EPS
Escola Politècnica Superior
UdG

Projecte/Treball Fi de Carrera

Estudi: Eng. Tècn. Informàtica de Gestió. Pla 2001

Títol: Catalogador de música MP3 y reproductor de música vía Web con búsquedas de música basadas en la definición de unas observaciones y emociones personales.

Document: Memoria

Alumne: José Emilio Navarro Rodríguez

Director/Tutor: Xavier Frigoler
Departament: Informàtica i Matemàtica Aplicada
Àrea:

Convocatòria (mes/any): Febrer/2007

Table Of Contents

1 Introducción

Hoy en día los usuarios de PC's domésticos guardan discotecas[1] de música MP3 (ver anexo 1) en los discos duros de sus PC's. En muchas ocasiones los usuarios tienen dificultades para localizar la música que desean escuchar. Para poder localizarla necesitarían una aplicación que permitiera catalogar sus discotecas musicales. Además es importante que la aplicación disponga de métodos de búsqueda eficientes: por artista, álbum, género musical o título de canción. No sólo se necesita catalogar y buscar sino también poder disfrutar de ella desde cualquier punto dónde se disponga de Internet.

Para personalizar el uso de las discotecas sería interesante que la aplicación dispusiera de las siguientes características: posibilidad de crear lista de reproducción (playlist, ver Anexo 3) y estadísticas de reproducción.

Las características descritas en los parágrafos anteriores son suficientes para usuarios con discotecas de poco volumen o sin necesidad de una explotación de datos exigente. En cambio para los usuarios con mayores discotecas y más exigentes, los recursos de las aplicaciones actuales son escasos y es necesario encontrar nuevas maneras de organizar y buscar música.

Según lo descrito anteriormente sería necesario desarrollar una aplicación con nuevas características de personalización y llegar más allá de los playlist y estadísticas. Para ello se propone crear dos nuevos tipos de información personal. Este tipo de información estará basada en el hecho de que la mayoría de los seres humanos recuerdan o asocian lugares, amigos con música . Además este "recuerdo" provoca en ocasiones un sentimiento o emoción.

Conocida el nuevo recurso de información los dos nuevos tipos de información serían:

- El primer tipo consistiría en información personal relacionada con amigos, lugares, comentarios, fecha, etc (observaciones personales).
- El segundo tipo estaría relacionado con “emociones personales” , rabia, felicidad, amor, etc que siente el usuario al escuchar una canción.

Este tipo de información sería posible relacionarla con las canciones de la discoteca previamente catalogada. Con las relaciones efectuadas la aplicación sería capaz de buscar música por nombres de amigos , lugares, ... o recomendar música en base a las emociones.

El proyecto que se va a desarrollar implementará las características descritas en los parágrafos anteriores, para cubrir las exigencias de los usuarios que buscan nuevas experiencias con sus discotecas. Además siguiendo la idea del proyecto, "personalización de discotecas", se complementarán las características mencionadas anteriormente con la posibilidad de puntuar los elementos musicales (canción, álbum, género, artista y playlist) .

Como soporte a las características de personalización la aplicación dispondrá :

- Un servicio que buscará enlaces web que contengan información/noticias de artistas

1 Colección de discos musicales o sonoros.

álbumes, etc.

- Una agenda musical en la cual los usuarios podrán programar el aviso de observaciones personales.
- Conexión con la base de datos pública (www.freedb.org) para normalizar la información obtenida de las discotecas de música MP3.

2 Objetivos

2.1 Objetivos del proyecto

El objetivo principal del proyecto es romper las fronteras de las búsquedas de canciones en las discotecas de los usuarios domésticos guardada en los discos duros de sus PC's. Para llevarlo acabo se quiere desarrollar dos nuevos tipos de búsquedas. El primer tipo buscará canciones en base a unas observaciones personales (amigos, lugares, comentarios, ...) que habrán sido definidas previamente por el usuario. El segundo tipo buscará canciones en base a las "emociones personales" que también serán definidas por el usuario.

2.1.1 Objetivos Secundarios:

- Organizar automáticamente los archivos de audio MP3 almacenados en los ordenadores personales.
- Reproducir los archivos de audio mediante Streaming (ver Anexo 4).
- Crear playlist (ver Anexo 3)personalizados.
- Permitir crear y asociar/vincular canciones, álbumes, artistas, playlists y géneros musicales (*elementos musicales*[2]) con "emociones personales".
- Definir criterios de búsqueda para las "emociones personales" ([3]tags).
- Poder seleccionar un *tag* a modo de estado/emoción personal y recuperar los elementos musicales asociados a las características del *tag*.
- Definir unas observaciones con información personal del usuario. Estas podrán ser relacionadas con elementos musicales y con ello poder realizar búsquedas a partir del nombre de un amigo o de un lugar concreto.
- Búsquedas a partir de los nombres de canciones, álbumes, artistas y géneros musicales. Han de ser rápidas y coherentes.
- Permitir representar el grado de satisfacción de un elemento musical mediante una puntuación, valores: entre 1 y 5.
- Mostrar estadísticas que informen de las 10 canciones mas reproducidas por el usuario actual y el total de usuarios de la aplicación. Canciones y álbumes con más puntuación. Los nuevos álbumes de la aplicación (top 10).
- Informar al usuario de la última vez que escucho un elemento musical.

2 Término será utilizado durante el desarrollo de la documentación para identificar a canciones, álbumes, artistas, playlists y géneros.
3 Término será utilizado durante el desarrollo de la documentación para identificar las "emociones personales".

- Proporcionar una herramienta para buscar enlaces con información/noticias de artistas, álbumes, ... sin tener que abandonar la aplicación.
- Organizar una agenda para recordar observaciones personales.
- Almacenar a modo de favoritos (*bookmarks*) las canciones preferidas de un usuario.
- Aplicación fácil de instalar en un PC de uso doméstico.
- La navegación de la aplicación tiene que ser cómoda e intuitiva.
- A nivel de desarrollo la aplicación tiene que ser el máximo de escalable y modular, simulando el trabajo en equipo.

2.2 Objetivos personales

- Escoger y utilizar un lenguaje de programación de servidor web orientado a objetos. Conocer sus ventajas y sus desventajas.
- Aprender a instalar y configurar el servidor web asociado con el lenguaje de programación escogido.
- Escoger una BDD libre (MySql, PostrgresSQL o Oracle Express). Conocer las ventajas y desventajas de utilizar una o otra.
- Utilizar una herramienta IDE (Integrated Development Environment).
- Escoger la metodología de desarrollo mas acorde para el proyecto y con ella utilizar las técnicas de ingeniería del software estudiadas.
- Utilizar los patrones de diseño de software estudiados. Especialmente el patrón MVC y el patrón DAO.
- Utilizar una herramienta de persistencia de objetos para implementar el patrón DAO.
- Conocer y estudiar la estructura y definición de los ficheros XML.
- Entender la estructura de los ficheros MP3 para poder recuperar la información que contienen sobre las canciones.
- Estudiar el acceso a las bases de datos musicales que son de uso libre para recuperar información de las canciones que tengan información sin normalizar.
- Estudiar y entender la emisión de datos vía streaming.

3 Características Generales

En este apartado se van a definir las características de la aplicación para poder solventar los problemas explicados en la introducción y respetar los objetivos del proyecto.

Debido a las diferentes y variadas características que va a tener el proyecto, estas se han definido en módulos independientes para facilitar su comprensión. Cada módulo tendrá sus requisitos. Al realizar estos módulos desde un principio facilitará el proceso de análisis y diseño que se realizará en capítulos posteriores. Además al realizar esta "modulación" puede simularse el trabajo en equipo ja que cada módulo podría formar parte de un equipo de desarrollo.

3.1 Usuarios del sistema

3.1.1 Usuarios normales

Toda persona que conozca la dirección dónde esta hospedado el servidor podrá darse de alta en el sistema. El usuario tendrá derecho a utilizar todas las funcionalidades que disponga la aplicación, excepto las reservadas para el administrador.

3.1.2 Administrador

Gestiona la música que se carga en el servidor. Es el encargado de añadir los álbumes a la aplicación. Para ello utiliza una funcionalidad especifica de la aplicación para definir la ruta de donde se encuentra el álbum a añadir.

3.2 Soporte MultiIdiomas

La aplicación permitirá escoger entre 3 idiomas diferentes: Catalán, Castellano e Ingles.

Por defecto será el Catalán.

3.3 Streaming

La aplicación permitirá reproducir mediante streaming cualquier elemento musical. El usuario seleccionará un elemento musical por ejemplo un álbum, automáticamente se generará un playlist (ver anexo 3) que habrá que descargarse en el computador. Una vez se dispone del playlist el usuario tendrá que ejecutarlo con un reproductor propio del PC, ya sea Winamp o Windows Media Player. Automáticamente el usuario empezara a escuchar la música servida por el servidor de streaming.

3.4 Catalogar música

El proceso de catalogar consistirá en indicar a la aplicación cual es la ruta local donde se encuentra la música, una vez definida comenzará un proceso automático:

- Primero se leerán y recuperara la información de los tags de los ficheros MP3.

- Segundo se eliminarán duplicados.
- Tercero y último se dará de alta en la base de datos la información obtenida.

Todo este proceso será mostrado por pantalla para informar al administrador de la evolución del mismo.

Aquellos archivos de audio que no tengan la información musical de los tags (ver Anexo 2) normalizada pasarán por un proceso de normalización. Este consistirá en obtener la información normalizada de la base de datos libre www.freedb.com. La nueva información normalizada será dada de alta en la base de datos.

3.5 Puntuación elementos musicales

Para conocer el gusto musical de los usuarios y con ello ayudar en las búsquedas de música, la aplicación dispondrá de un elemento que permitirá definir una puntuación, entre 1 (mínimo) y 5 (máximo). La puntuación será totalmente independiente entre elementos musicales, será possible tener un álbum con puntuación 5 pero algunas de sus canciones con puntuación 1 o sin puntuar.

3.5.1 Requisitos

- La puntuación se podrá realizar tantas veces como el usuario lo desee.
- La primera vez que se de una puntuación el valor será sumado a una puntuación genérica de la aplicación para poder realizar estadísticas.
- Hasta que un elemento musical no sea puntuado el valor es 0.

3.6 Observaciones/Recuerdos

Cada usuario podrá definir sus propias observaciones personales. Las características que definirán una observación serán: título de la observación, un lugar, unos amigos, una URL, una fecha y un comentario. Además será posible añadir una fotografía no superior a 1 Megabyte.

Una vez se haya definido una observación podrá ser asociada con más de un elemento musical.

Con las observaciones definidas y las asociaciones realizadas la aplicación será capaz de buscar elementos musicales basados en las características de una observación.

3.6.1 Mantenimiento

Cada usuario podrá dar de alta, modificar y eliminar sus observaciones.

3.6.2 Requisitos

- Las observaciones son personales de cada usuario.
- Un elemento musical tiene una única observación asociada.

- Una observación puede estar en mas de un elemento musical.
- Los elementos musicales que tengan asociada una observación serán identificados con un icono especial.

3.7 Emociones/*Tags*

Las emociones serán unos *tags* especiales que permitirán identificar que estado de ánimo nos hace sentir un elemento musical. La aplicación será capaz de buscar y seleccionar todos los elementos musicales identificados por un mismo *tag*.

Cada usuario será responsable de crear y caracterizar sus propios *tags*. La caracterización consistirá en definir el número de canciones máximas que se quieren escuchar, los elementos musicales sobre los cuales se realizará la búsqueda, una puntuación o un intervalo de puntuación y las fechas en que se hayan realizado las reproducciones.

3.7.1 Mantenimiento

Cada usuario podrá dar de alta, modificar y eliminar sus *tags*.

3.7.2 Requisitos

- La definición de los *tags* podrá variar tantas veces como el usuario desee.
- Los *tags* serán únicos de cada usuario.
- Los tags por defecto serán: Alegría, Tristeza, Rabia y Miedo.
- La aplicación ha de facilitar una herramienta para seleccionar un *tag* e iniciar la búsqueda de elementos musicales.
- El usuario ha de poder seleccionar los elementos musicales resultantes de la búsqueda.

3.8 PlayLists

Los playlists (ver anexo 3) permitirán al usuario agrupar canciones de modo que pueda localizarlas de manera más rápida y cómoda.

3.8.1 Mantenimiento

Alta

Cada usuario podrá crear tantos playlist como necesite. Un playlist constará de un nombre que lo identifica y un número indeterminado de canciones. Las canciones podrán ser añadidas al playlist cuando se muestre el resultado de una búsqueda o en el detalle de álbumes donde se mostrarán las canciones.

Modificar

Una vez dado da alta un playlist se podrá cambiar el nombre, añadir y quitar canciones sin

restricciones.

Eliminar

En todo momento se podrá eliminar un playlist.

3.8.2 Restricciones

- Cada usuario es propietario de los playlist que el crea.

3.9 Búsquedas

La aplicación dispondrá de un módulo en el cual el usuario seleccionará el campo de búsqueda (nombre de: álbum, canción, amigo, lugar,..) y esta devolverá un resultado con las coincidencias del nombre. Del resultado obtenido el usuario podrá escoger que elementos desea reproducir.

Los tipos de búsqueda estarán catalogados en dos grupos:

3.9.1 Uso común

Este tipo de búsqueda consistirá en introducir el nombre de un álbum, canción, artista o género y recuperar todos los elementos musicales del mismo tipo que contengan ese nombre.

3.9.2 Personalizadas

Estas búsquedas se basaran en las observaciones y los tags.

- *tags:* Búsquedas tal como se han detallo en el punto 3.7.
- Observaciones: Buscar todos los elementos musicales que tengan asociada una observación con el nombre de una persona, un lugar, una url, una fecha o parte de un comentario.

3.10 Estadísticas

Para facilitar al usuario a escoger música se va a recoger información sobre cuatro tipo de variables:

- La primera informará de las canciones y artistas con más puntuación.
- La segunda informará de las canciones más reproducidas, tanto para el usuario activo como el total de usuarios del sistema.
- La tercera variable consistirá en informar de los nuevos álbumes que han sido añadidos al catalogo.
- La cuarta y última informará de la última vez que fue reproducido un elemento musical.

3.10.1 Requisitos

- Almacenar el número de reproducciones de los elementos musicales para cada usuario y el conjunto de usuarios.
- Histórico de reproducciones, con el cual se podrá saber la última y primera reproducción.
- Fecha de carga de los álbumes en el sistema.

- La información de las variables será mostrada en el home en formato top 10.

3.11 Agenda musical

La agenda musical permitirá definir observaciones que quieren ser recordadas (eventos). En caso de que la observación tengan elementos musicales asociados podrán ser seleccionadas para su reproducción.

3.11.1 Mantenimiento

Cada usuario podrá dar de alta, modificar y eliminar los eventos de su agenda personal.

3.11.2 Restricciones

- Todo evento tiene que tener definida una observación activa y una fecha.
- Los días que haya eventos aparecerá un aviso en la pagina principal.
- Los eventos de fechas inferiores a la actual serán eliminados automáticamente.

3.12 Buscador de Links

La aplicación dispondrá de una herramienta que permitirá acceder al buscador google sin que el usuario tenga que abandonar la aplicación. Esta estará formada por dos tipos de búsquedas:

- Búsqueda de páginas web mediante servicios web (ver capítulo 6.7).
- Búsqueda de noticias RSS (ver capítulo 6.8).

3.12.1 Restricciones

- [4]Los resultados de las búsquedas no serán superiores a 10 links.

3.13 Canciones Favoritas

Esta característica permitirá marcar como favoritas las canciones que el usuario desee y así tener un acceso mas rápido a dichas canciones.

3.13.1 Mantenimiento

Alta

Cada usuario podrá añadir canciones a su lista de favoritas.

El alta será posible desde diferentes puntos de la aplicación donde aparezcan canciones. Por ejemplo en el detalle de un álbum o desde la vista de los top 10.

4 Google únicamente ofrece 10 links de forma gratuita.

Eliminar

En todo momento se puede eliminar una canción de la lista de favoritas.

3.13.2 Restricciones

- Se guardará el día que se ha dado de alta como favorita una canción.

4 Estudio De Metodologías

A continuación se realizará un estudio de las diferentes metodologías de trabajo disponibles para el desarrollo del proyecto. Una vez finalizado el estudio escogeremos la mejor opción para el desarrollo del proyecto.

4.1 Metodologías disponibles

Les metodologías que van ha ser estudiadas son:

- Métrica Versión 3
- Proceso unificado (o Rational Unified Process)
- Iconix

4.1.1 Métrica Versión 3 (MV3)

Analizando esta primera metodología, podemos decir que la MV3 es una metodología orientada a objetos, su principal característica es que en cada momento se tiene que adaptar y dimensionar según las características de cada proyecto. La MV3 divide cada uno de los procesos en actividades, y estos en tareas, por cada tarea se describen las principales acciones, productos y técnicas utilizadas. Algunos de sus inconvenientes pueden ser: problemas con el mantenimiento del software, mala compresión de los requisitos del cliente y una calidad de software no muy buena. Es una metodología muy estricta, ya sea en su utilización como en en la generación de documentación. Actualmente esta metodología es utilizada únicamente a nivel nacional, especialmente en los proyectos de la administración pública española.

4.1.2 Proceso unificado (o Rational Unified Process,RUP)

Esta segunda metodología es también orientada a objetos, este hecho permite que el proceso de desarrollo sea más flexible, con lo cual se puede adaptar y modificar mejor el proyecto según sus características. El RUP se puede definir como un método de desarrollo iterativo e incremental, este hecho permite que el proyecto se pueda organizar en pequeños proyectos de duración mas corta llamados iteraciones. Cada iteración tiene sus propias actividades de análisis de requerimientos, diseño, implementación y la correspondiente fase de pruebas. El RUP realiza un desarrollo en paralelo entre documentación y arquitectura con ello se minimiza la repetición de tareas y permite un incremento en la reutilización de componentes y una mejor facilidad en el mantenimiento del software. Existen hasta 4 fases diferentes en esta metodología:

- **Iniciación:** Se realiza una planificación del proyecto, incluye una estimación de los recursos necesarios.
- **Elaboración:** Los objetivos son analizar cual es el dominio del problema y establecer cual será la arquitectura correcta, marcar los objetivos del sistema, y por último se

decide si se pasa a la fase de construcción.

- **Construcción:** Durante esta fase se desarrolla el software de manera iterativa e incremental. Este hecho incluye añadir los requisitos que no fueron descubiertos al principio, refinamiento del diseño y las pruebas de software.
- **Transición:** En esta fase se realiza la implantación del software en las maquinas de los usuarios.

Al finalizar el proceso en algunos casos aparecen algunos ajustes necesarios en el software. En la mayoría de los casos esta fase empieza con una versión beta, que posteriormente y una vez se han realizado los cambios necesarios será la versión final.

Una de las diferencias con la MV3 es que el RUP tiene un reconocimiento internacional y es una metodología más utilizada en las empresas.

4.1.3 Iconix

La tercera metodología es utilizar Iconix, se podría definir como una metodología basada en el RUP, pero con la diferencia que esta es una versión más simple y con algunas mejoras con respecto al RUP. Como el RUP, Iconix es una metodología orientada a objetos, se realiza de forma iterativa e incremental. Existen tres fases diferentes durante el desarrollo: definición de requerimientos, análisis y diseño orientado a objetos utilizando la notación UML.

En la primera fase se realiza el modelo de casos de uso que definirá las necesidades del sistema, los casos de uso vendrán especificados por su ficha de casos de uso y si es necesario se adjuntará el diagrama de actividades. En algunos casos también se facilita un prototipo de la interfaces de usuario.

En la fase de diseño se buscará la solución para poder implementar las soluciones de requerimientos del sistema. Se pueden diferenciar dos visiones diferentes: **la dinámica,** representa las colaboraciones entre objetos mediante los diagramas de secuencia, por la otra banda **la estática**, se ve representada por el diagrama de clases de diseño.

4.2 Alternativa seleccionada

Una vez presentadas las diferentes metodología Iconix ha sido la escogida.

El motivo de su elección ha sido principalmente porque es una metodología ágil y clara. Este hecho hace que se adapte perfectamente al proyecto, ya que el mismo estará compuesto por varios módulos cuya carga no será excesiva y no requieren de documentación excesiva.

Si analizamos la MV3 esta contiene muchas fases y seria excesiva por sus características, esta es mas propia de proyectos más grandes y más complejos. Con respecto al RUP al igual que la MV3 esta formado por demasiadas fases. Ademas requieren especificar todos los procesos realizados y con ello generar una documentación excesiva para el tipo de proyecto que se va a desarrollar.

5 Análisis Y Diseño

Una vez se han presentado las característica generales de la aplicación a desarrollar y se conoce la metodología para desarrollar el proyecto (Iconix) ya se puede comenzar el proceso de análisis.

Primero se van a describir los casos de uso, con ello conseguiremos conocer cuales son las situaciones que nos podemos encontrar y cuales son las funcionalidades de la futura aplicación. Para describir el escenario principal y alternativos se utilizaran las fichas de caso de uso. En los casos que sea necesario se realizaran los diagramas de actividad para dar mayor claridad a los aspectos más complicados del caso de uso.

En segundo lugar se realizará un prototipo de interfaces de usuario en aquellos casos que se crea conveniente.

En tercer lugar a partir de los requerimientos planteados anteriormente, se procederá a realizar los diagramas de colaboración simplificados en los casos que no queden claras las fichas de caso de uso. Con el diagrama se observará como interactuan los diferentes objetos de las clases.

El cuarto y último punto consistirá en identificar los mensajes que se tienen que pasar entre los objetos, y entre los objetos y las operaciones, en el diagrama de secuencia quedaran reflejados todos estos aspectos.

5.1 Diagramas de casos de uso (CU)

A continuación se van a presentar los actores y casos de uso.

Tal como se ha comentado en el apartado de características generales de la aplicación se va a realizar una agrupación por módulos para facilitar la comprensión y identificación de las funcionalidades que corresponden a cada uno.

5.1.1 Actores

- Administrador (Admin)
- Usuario corriente (UC)
- Sistema.
- Reproductor (RE)

5.1.2 Usuarios

- Registrar usuario (UC)
- Modificar datos (UC)

5.1.3 Soporte Multiidioma

- Seleccionar palabra de un idioma.(Sistema)

5.1.4 Streaming

- Reproducir Elemento Musical.(UC)
- Crear PlayList para descargar. (Include reproducir Elemento Musical)
- Reproducir PlayList Descargado.(RE)

5.1.5 Catalogar música

- Dar de alta música de un directorio (Admin)
- Buscar información en base de datos libre. (Sistema)

5.1.6 Puntuación elementos musicales

- Puntuar elemento musical. (UC)
- Seleccionar Elemento Musicales(Include de Puntuar)

5.1.7 Observaciones

- Mantener observaciones (Alta,Eliminar y Modificación) (UC)
- Asociar observación con elemento musical.(UC)
- Obtener observaciones Usuario.(Incluido por los dos anteriores)

5.1.8 Tags

- Mantener tags (Alta,Baja y Modificación) (UC)
- Vincular *tag* con elemento musical (UC)

5.1.9 PlayList

- Mantener playlist (Alta,Eliminar y Modificación) (UC)
- Añadir/Eliminar canciones a un playlist. (UC)

5.1.10 Búsquedas

- Buscar canciones por nombre álbum, canción, artista o género(UC)

- Reproducir elementos resultantes de la búsqueda. (UC)
- Buscar observaciones. (UC)
- Buscar elementos musicales de una observación.(UC)
- Reproducir música observación.(UC)
- Buscar música vinculada Tag.(UC)
- Reproducir resultados T*ag*. (UC)

5.1.11 Estadísticas

- Mostrar las 10 canciones más escuchadas por usuario. (Sistema)
- Mostrar las 10 canciones más escuchadas por total usuarios. (Sistema)
- Mostrar las 10 canciones más puntuadas. (Sistema)
- Mostrar los 10 artistas con más puntuación. (Sistema)
- Mostrar los 10 últimos álbumes añadidos al sistema. (Sistema)
- Informar de la última fecha de reproducción de un elemento musical. (Sistema)
- Mostrar contador reproducción elemento musical. (Sistema)

5.1.12 Agenda musical

- Mantener eventos agenda musical (Alta,Eliminar y Modificación). (UC)
- Avisar eventos. (Sistema)
- Ver próximos eventos.(Sistema)
- Reproducir música eventos.(UC)

5.1.13 Canciones Favoritas

- Mantener canciones favoritas (Alta y Eliminar). (UC)

5.1.14 Buscador de páginas web

- Buscar páginas web sobre un artista.(UC)
- Buscar noticias web de un artista. (UC)

A continuación se presentan los escenarios principales separados por actor (*usuario,administrador y sistema*). El escenario del usuario tiene un gran número de casos de uso, por ello se ha decidido de dividirlo en dos y facilitar a si su comprensión.

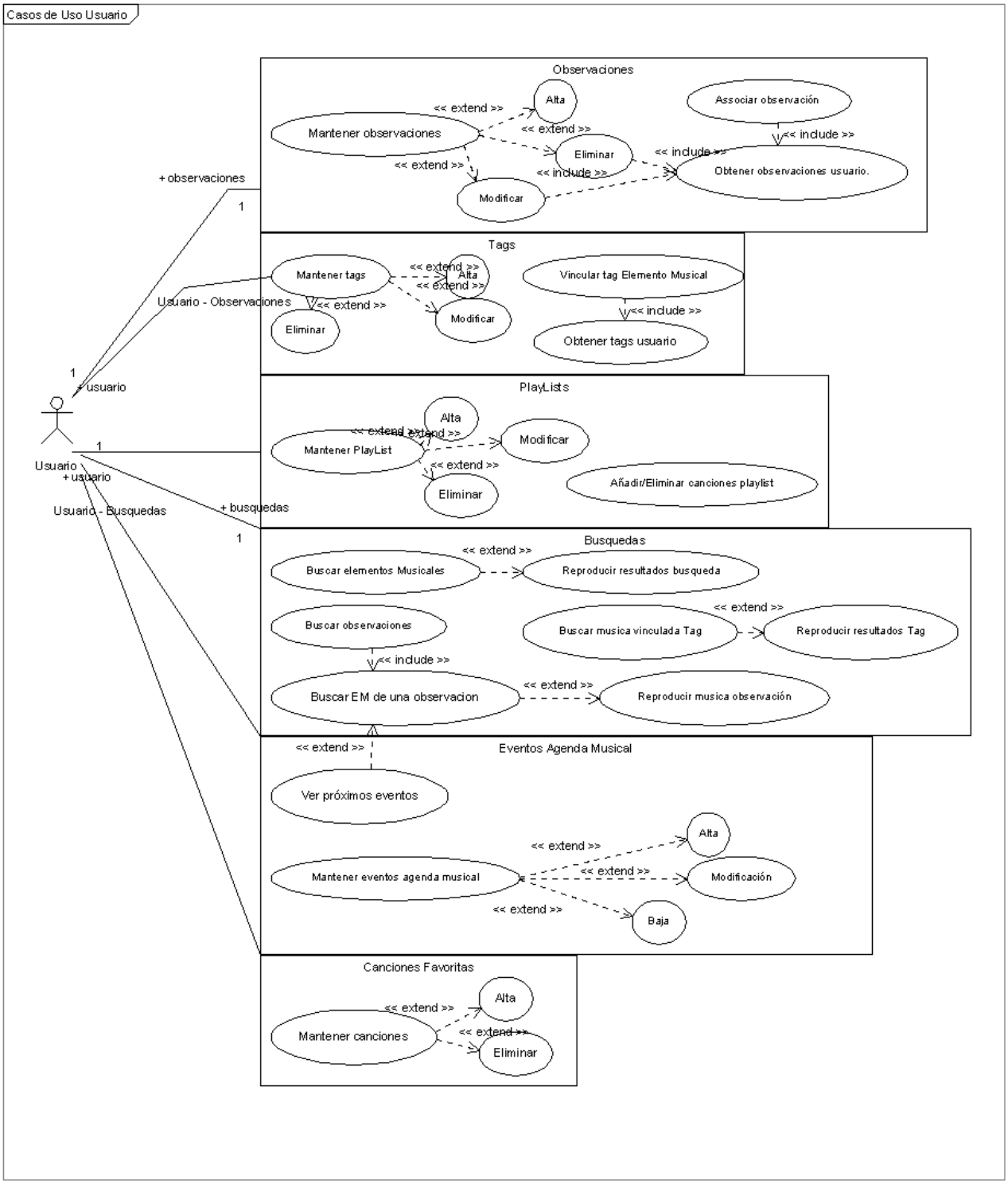


FIG 1. Escenario principal del USUARIO Parte 1

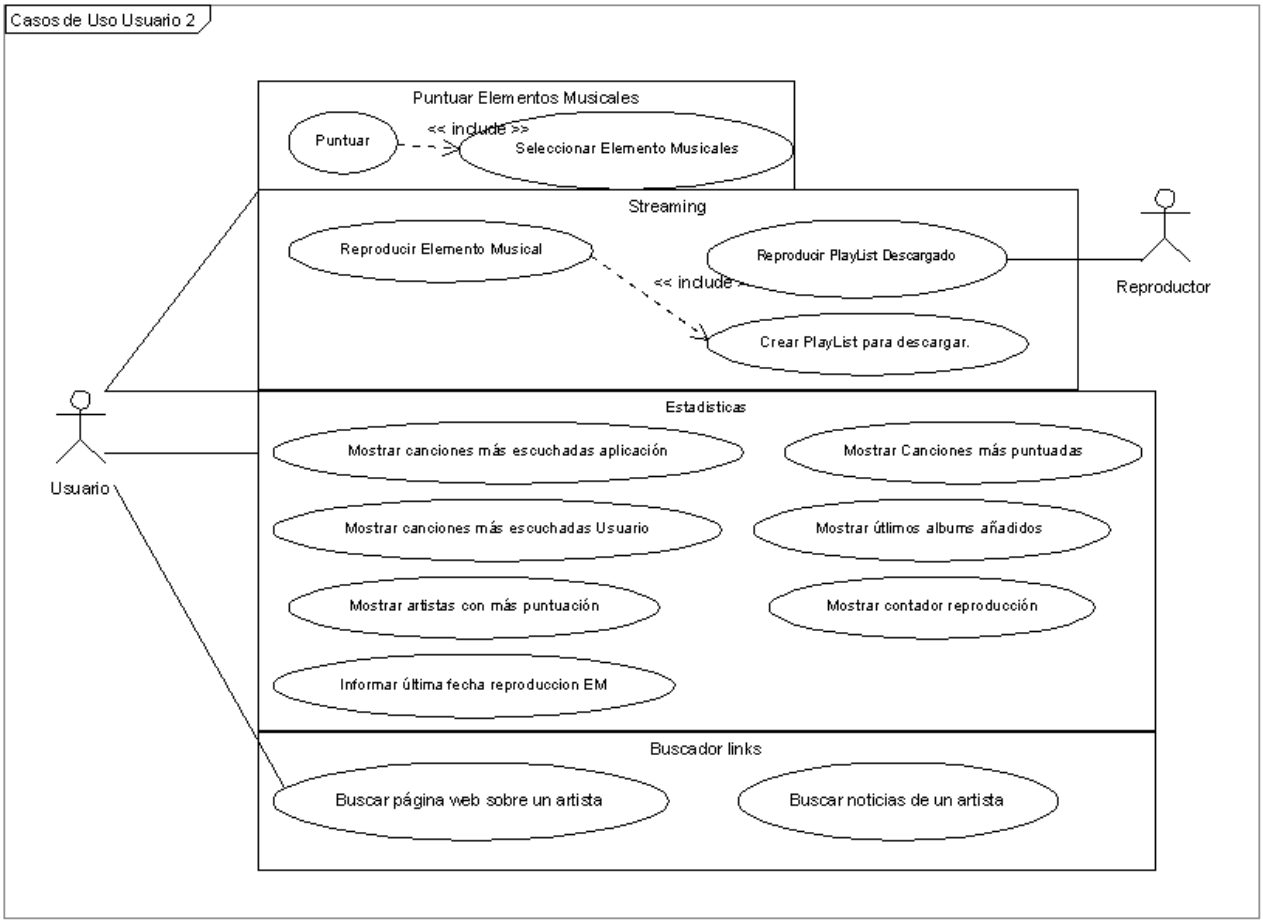


FIG 2: Escenario principal USUARIO Parte 2

En el escenario de la FIG 2 se ha añadido el actor *Reproductor* y su único caso de uso. Al ser sólo uno y estar directamente relacionado con el módulo *Streaming* se ha decidido no separarlo.

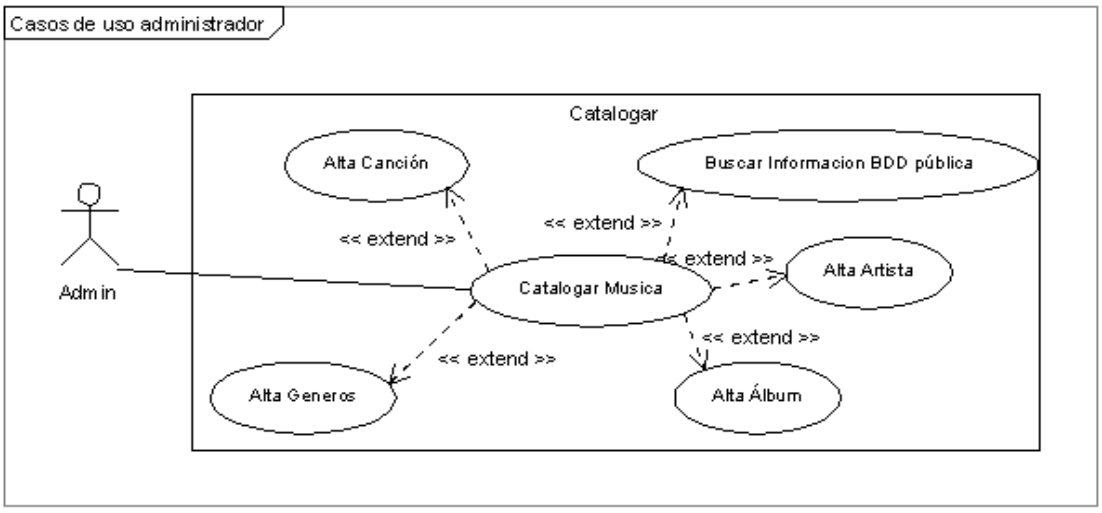


FIG 3: Escenario principal ADMINISTRADOR

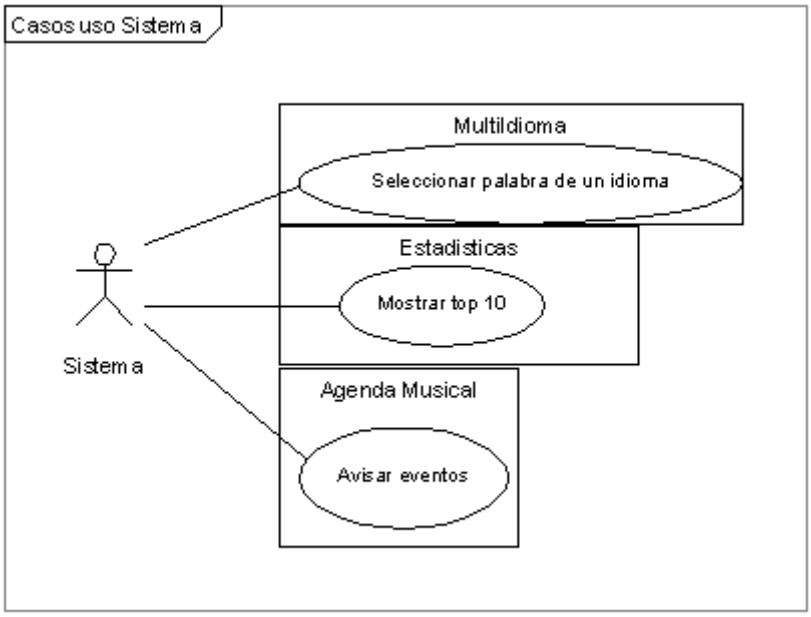


FIG 4.Escenario principal SISTEMA

Como se puede comprobar en las figuras, la organización de los casos de uso se ha estructurado en paquetes, para facilitar la gestión, con ello en posteriores mejoras y modificaciones será más fácil su uso.

5.2 Aclaraciones previas de los casos de uso.

Antes de continuar con el análisis y diseño de los casos de uso, es necesario entender como se ha estructurado la documentación y el significado de los estereotipos utilizados.

La metodología Iconix prácticamente une el análisis y el diseño en uno, por eso se han unido en el mismo capítulo. Con ello se consigue facilitar la comprensión del caso de uso y conseguir en el momento de la programación un desarrollo con un valor añadido de calidad. Por lo tanto tendremos una introducción para cada caso de uso que informará del módulo de programación al cual pertenece (ver el capítulo *Tecnologías utilizadas, aplicación del patrón MVC* para conocer la importancia del módulo). Seguido de la introducción vendrá la ficha de casos de uso, el prototipos de interfaces, sus correspondientes diagramas y las explicaciones para dejar el caso de uso más claro.

Los casos de uso donde el nivel de complejidad no sea elevado y sean similares (como son los Mantenimientos) únicamente se adjuntará la ficha de casos de uso, ya que si seguimos la metodología Iconix, esta especificar que solo se adjuntaran los elementos que sean necesarios para clarificar y detallar las funcionalidades.

Los casos de uso van a implementar el patrón de comportamiento MVC (Model View Control). El uso del patrón facilitará la representación en los diagramas de colaboración y secuencia y por supuesto en la codificación. En el capítulo 7.4 se darán mas detalles de este patrón y de otros patrones utilizados.

Estereotipos utilizados en los diagramas de secuencia y colaboración:

- **Control:** Son las clases de control utilizadas en cada módulo.
- **TPL:** Son las interfaces con las cuales interactuar el usuario, se han llamado TPL porque es la abreviación de *template*. *Templates* son las plantillas utilizadas para construir las pantallas, ver capítulo *Tecnologías utilizadas, Smarty* (capítulo 6.10).
- **Actor:** Es el actor del sistema que inicia el caso de uso.
- **MultiObjeto:** Representa una colección de un tipo de objetos.
- **Arquitectura:** Identifica las clases que forman parte de la arquitectura del sistema.
- **Archivo:** Representar archivos físicos del servidor.

Clases especiales de los diagramas de secuencia y colaboración:

- **DB_<nombre colección objetos>**: Las clases modeladas con este nombre representan la persistencia de objetos en la base de datos y memoria. Ejemplo de definición: DB_Users.
  Para mas información de las técnicas de persistencia utilizadas ver capítulo 6.4.

5.3 Diagrama de clases

Para poder entender mejor los casos de uso a continuación se presenta el diagrama de clases con los principales objetos.

Del diagrama hay que destacar dos objetos como los más importantes, El objeto *Song* que tiene una relación directa con los objetos *Album, Genere, Artist y PlayList*. Estos cuatro objetos más *Song son* una generalización del objeto MusicElement. Es muy importante este objeto ya que en todos los casos de uso se utilizará. También en las explicaciones de los diagramas. Con ello conseguimos simplificar los diagramas y las explicaciones.

El segundo objeto importante es *UserUsePlay* este almacena información del uso de los elementos musicales, observaciones y emociones. El uso puede ser: el número de reproducciones de elementos musicales, la puntuación que cada usuario da a un elemento musical, las observaciones que un usuario asocia y las emociones vinculadas a elementos musicales.

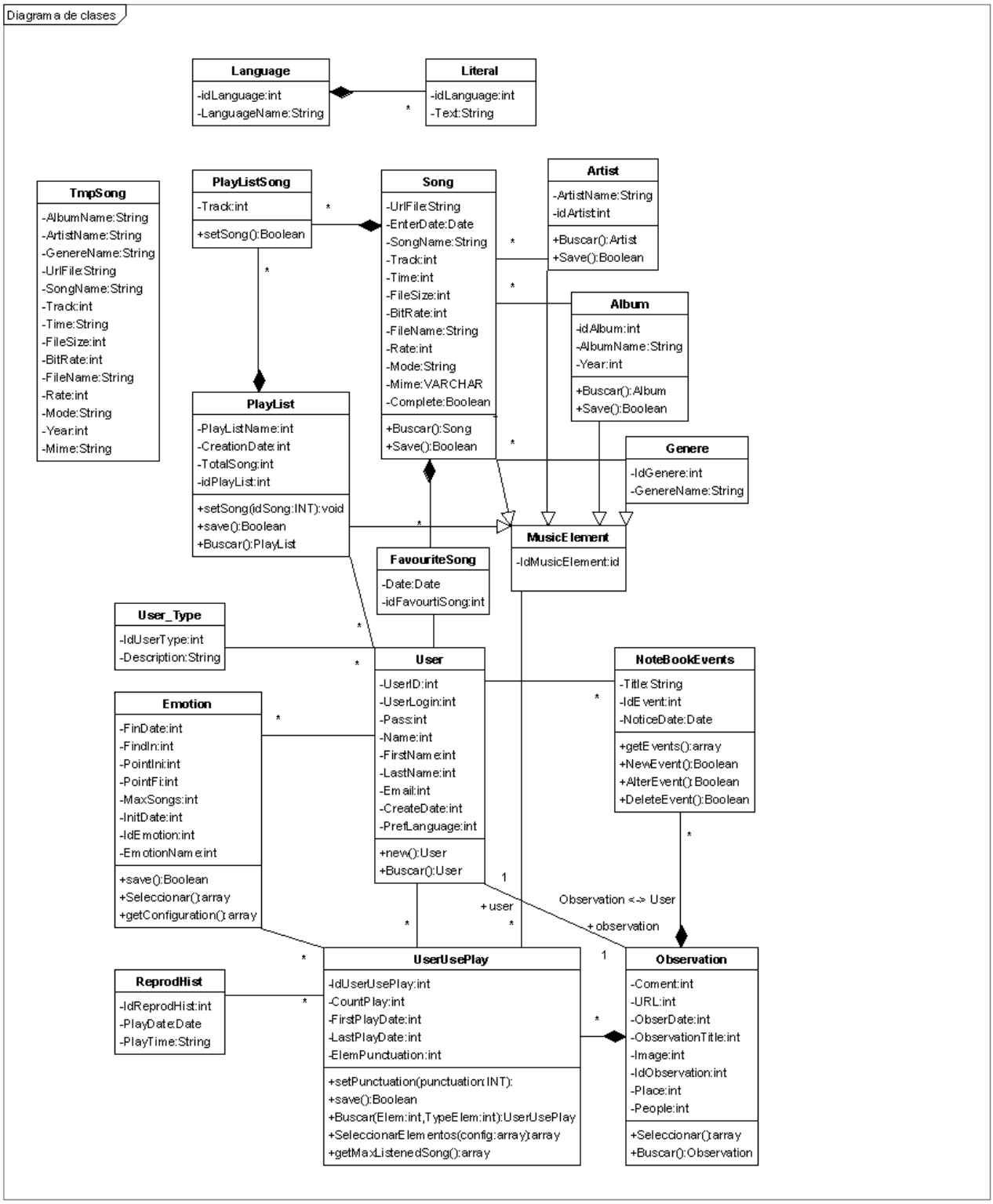


FIG 5: Diagrama de clases

5.4 Registrar Usuario

Este caso de uso forma parte del módulo: **Users.**

El proceso de registrar usuario lo llevará acabo el propio usuario. Cuando se conecte a la URL de la aplicación aparecerá la pantalla de entrada al sistema donde se encontrará la opción de registrarse en el sistema. Una vez registrado correctamente podrá entrar en él.

5.4.1 Ficha de caso de uso

CASO DE USO	***Registrar Usuario***		
Versión	2	**Fecha**	03/05/06
Descripción	Se registra un usuario en el sistema, para poder acceder posteriormente		
Actores	Usuario		
Precondición	Cierto		
Flujo principal	1 Registrar usuario 1.1 Entrar Datos (Nombre, primer apellido, segundo apellido, email, identificación entrada, contraseña y idioma uso) 2 Añadir 2.1 Crear 2.2 Añadir usuario 2.3 Añadir emociones/tags estandards		
Flujos alternativos	En caso de no entrar correctamente un dato se muestra un mensaje de error con el campo incorrecto.		
Postcondición	Usuario registrado en el sistema y esta autorizado a entrar en él.		
Requisitos	1 Crear directorio para las fotos de las observaciones. El nombre del directorio será el identificador único que ha generado la BD.		
Comentarios	El identificador único de usuario y la fecha de creación se generaran y guardaran automáticamente .		

El punto 2.3 cumple los requisitos especificado en capítulo 3.7.2. Los valores iniciales del punto 2.3 serán definidos con los parámetros estandards de la aplicación, una vez el usuario este dentro podrá cambiarlos según sus preferencias. Más detalles ver caso de uso *Mantener tags*

5.4.2 Prototipo interface

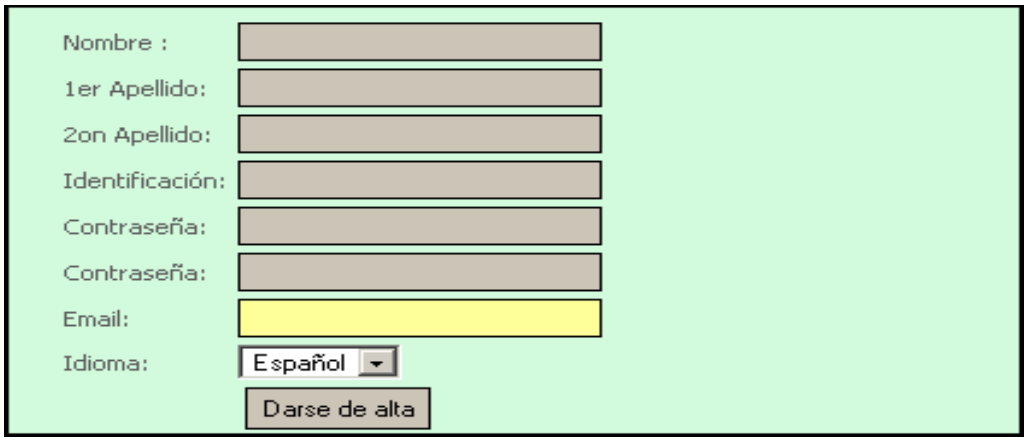


FIG 6: Interface

5.4.3 Diagrama de colaboración simplificado

Del siguiente diagrama hay que destacar la navegación del usuario. Una vez se han introducidos los datos en la pantalla "AdminUsers" y el proceso no ha generado ningún error se muestra la pantalla inicial que permite al usuario acceder al sistema.

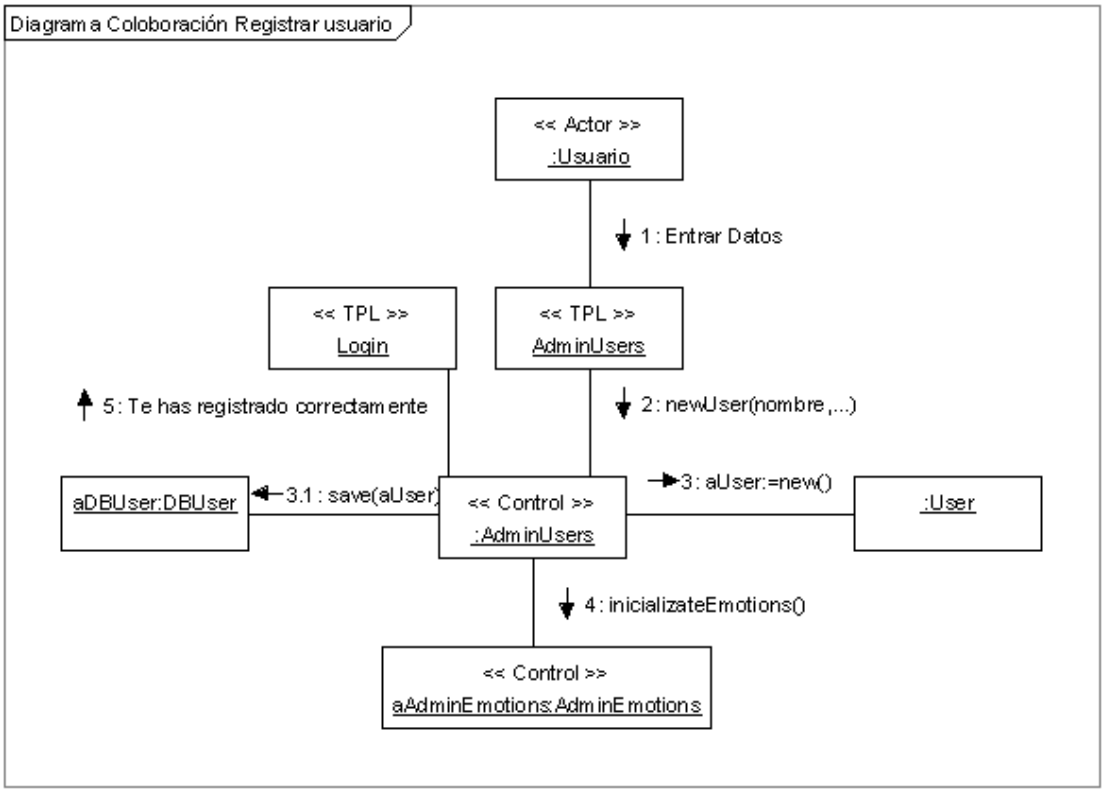


FIG 7. Diagrama de colaboración

Respecto al caso de uso detalle de alta de emociones iniciales,(4 inicializateEmotions()) es el mismo que el caso de uso alta emociones con la diferencia que los datos son proporcionados por el sistema y no por el usuario.

5.4.4 Diagrama de secuencia simplificado

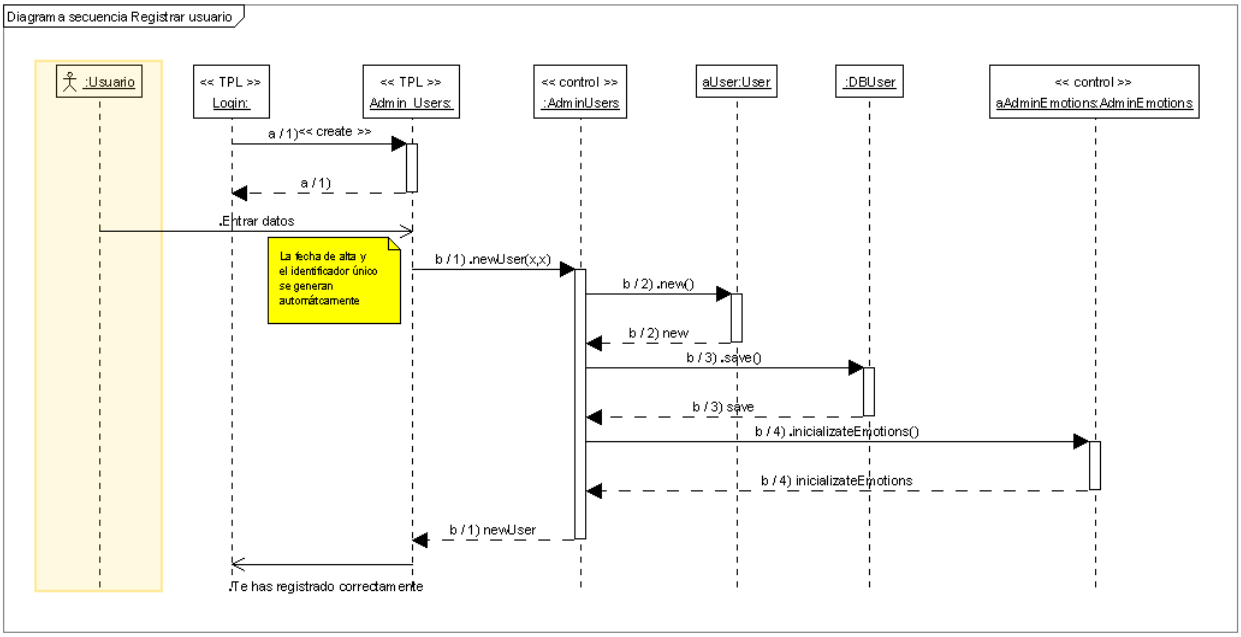


FIG 8. Diagrama de secuencia

Al igual que el diagrama anterior el detalle de alta de emociones iniciales,(inicializateEmotions()) es el mismo que el caso de uso alta emociones.

5.5 Modificar datos usuario

Este caso de uso forma parte del módulo: **Users.**

El usuario una vez esta dentro del sistema tiene la posibilidad de modificar sus datos personales y el idioma de navegación. A través del menú principal de la aplicación tendrá acceso a sus datos, una vez los haya modificado se mantendrá en el misma pagina para que pueda validar los cambios.

La interface tiene los mismos atributos que el caso de uso *Registrar Usuario (5.4)*

5.5.1 Ficha de caso de uso

El caso de uso tiene dos flujos principales, el primero se encarga de recuperar los datos actuales y mostrarlos al usuario. El segundo modifica los datos nuevos que ha entrado el usuario y muestra los cambios.

CASO DE USO	***Modificar datos usuario***		
Versión	2	**Fecha**	03/05/06
Descripción	Permite modificar los datos del usuario actual del sistema.		
Actores	Usuario		
Precondición	El usuario ha entrado al sistema		
Flujo principal	1 Recuperar datos usuario 1.1 Buscar usuario por identificador único. 1.2 Mostrar datos 2 Entrar Datos 2.1 Buscar usuario por identificador único. 2.2 Modificar Datos(Nombre, primer apellido, segundo apellido, email, identificación entrada, contraseña y idioma uso) 2.3 Actualizar datos usuario 2.4 Actualizar variable de sesión idioma 2.5 Mostrar datos actualizados		
Flujos alternativos	En caso de no entrar correctamente un dato se muestra un mensaje de error con el campo incorrecto.		
Postcondición	Se han alterado los campos del usuario activo en el sistema.		
Comentarios	El identificador único lo proporciona la sesión actual.		

5.5.2 Diagrama de colaboración simplificado

Para no complicar la representación y entender mejor el caso de uso se ha dividido en dos diagramas de colaboración. El primero muestra como recuperar los datos a modificar (FIG 9) y el segundo como se realizan los cambios (FIG 10).

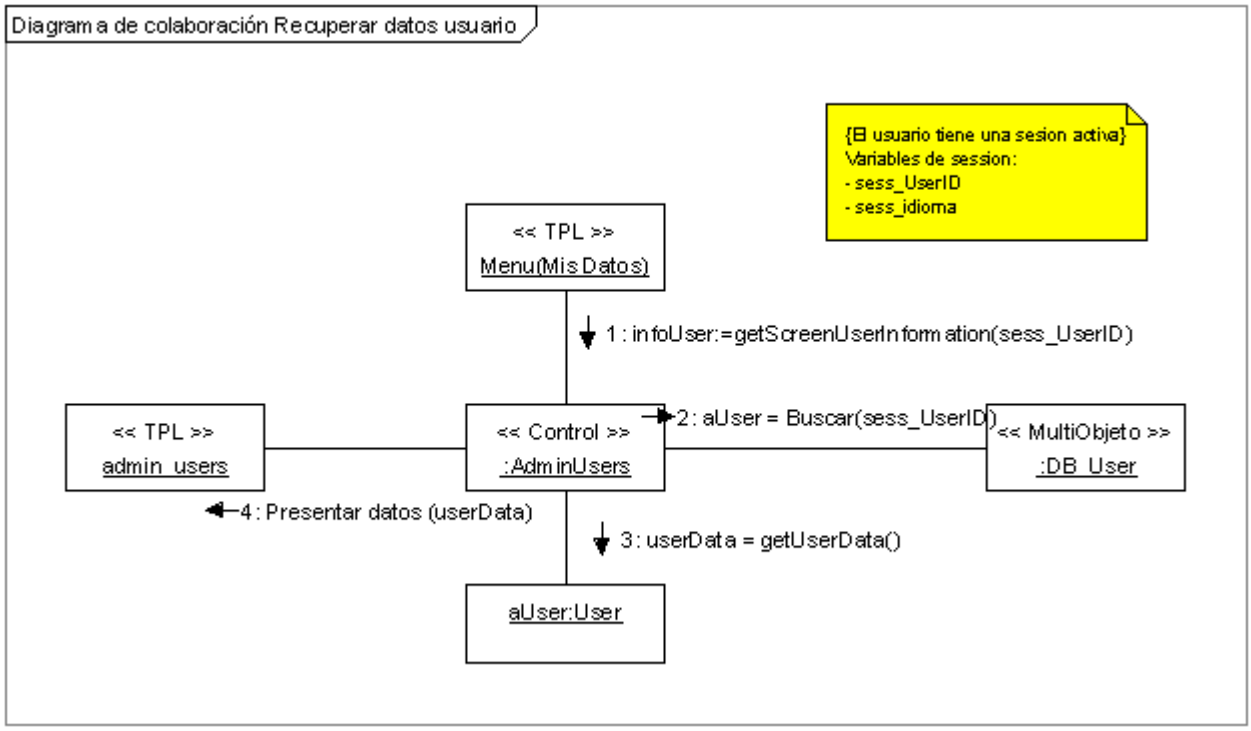


FIG 9: Diagrama de colaboración

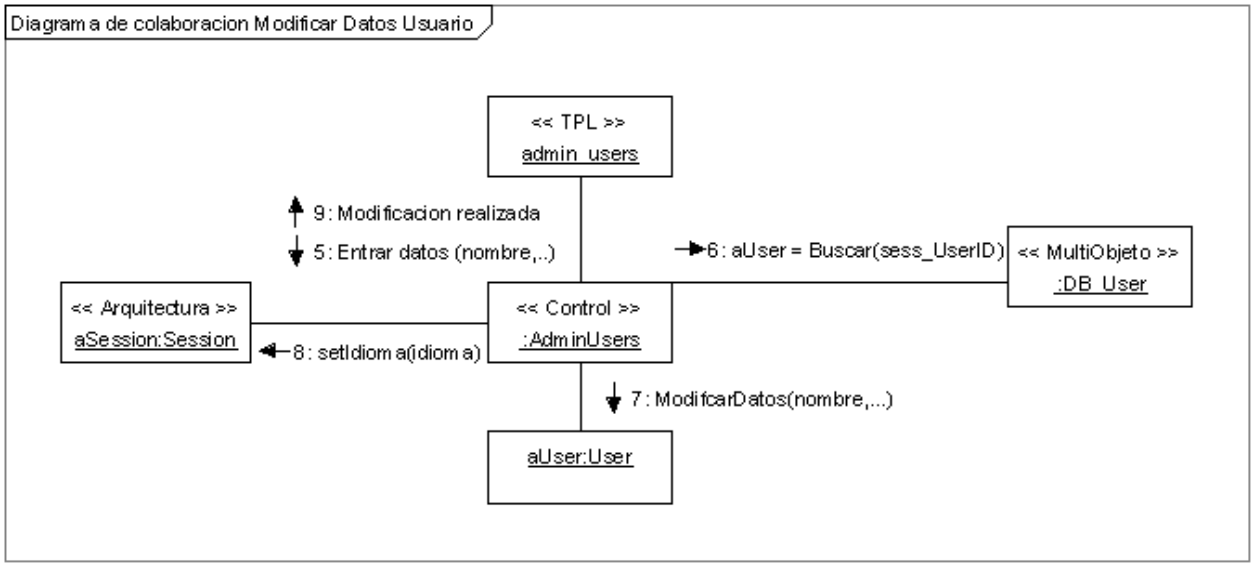


FIG 10. Diagrama de colaboración

5.5.3 Diagrama de secuencia simplificado

No se ha creído necesario diseñar este diagrama ya que los diagramas de colaboración anteriores representar la interacción de los objetos con claridad.

5.6 Seleccionar palabra de un idioma

Este caso de uso forma parte del modulo:**Arquitectura.**

Este caso de uso se encarga de seleccionar la palabra correspondiente al idioma preferente de usuario activo.

5.6.1 Ficha de caso de uso

CASO DE USO	***Seleccionar palabra de un idioma***		
Versión	1	**Fecha**	14/06/06
Descripción	Recupera una palabra coincidentes con el idioma del usuario activo		
Actores	Sistema		
Precondición	Idioma preferente seleccionado por el usuario		
Flujo principal	1 Seleccionar palabra 1.1 Obtener Idioma 1.2 Obtener identificador palabra 1.3 Acceder a los datos 2 Mostrar literal		
Flujos alternativos			
Postcondición	Se muestra la palabra con el idioma preferente del usuario activo		
Comentarios			

5.6.2 Prototipo interface

Al ser un proceso interno del sistema no requiere interface. El resultado esta presente en todas las interfaces de la aplicación.

5.6.3 Diagrama de colaboración simplificado

No se ha diseñado.

El diagrama de secuencia aporta toda la información necesaria para comprender la secuencia de mensajes.

5.6.4 Diagrama de secuencia simplificado

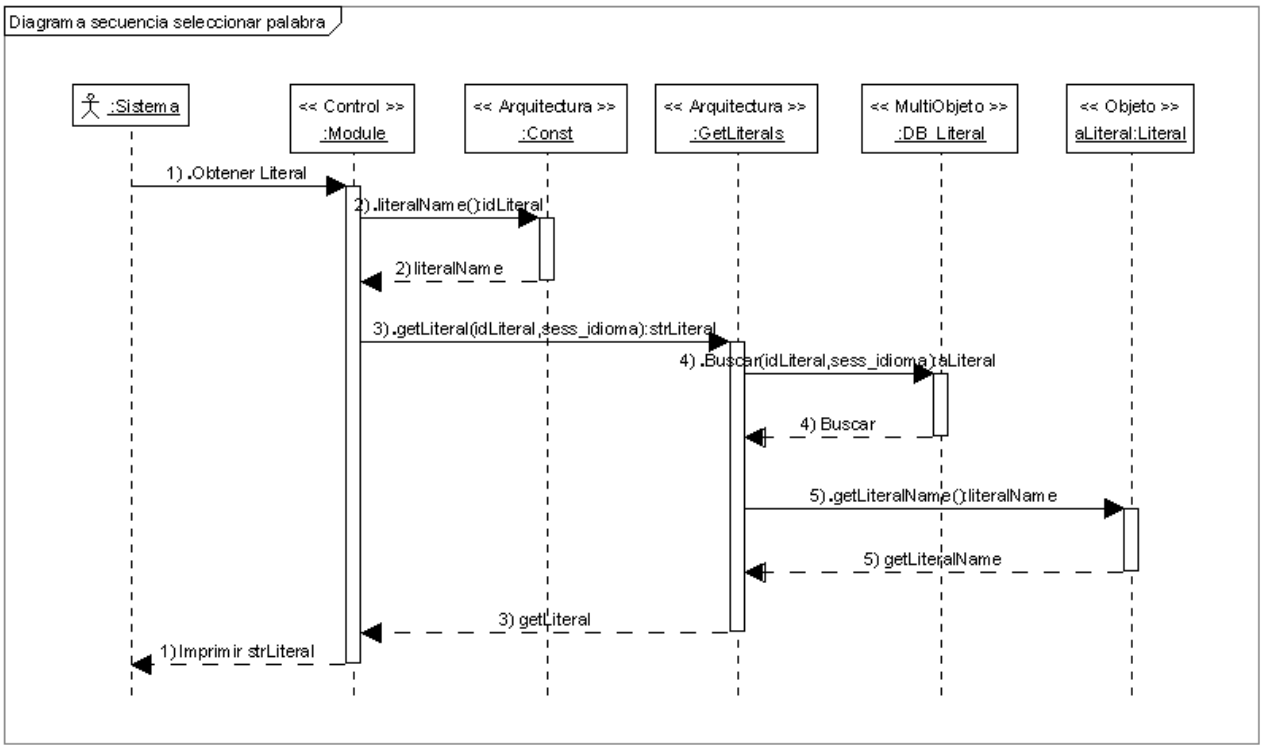


FIG 11: Diagrama de secuencia

5.7 Reproducir elemento musical

Este caso de uso forma parte del módulo: **Streaming.**

Recordar como ya se ha especificado anteriormente, se considera un elemento musical (EM): una canción, un álbum, un artista y un playlist, ya que todos ellos tiene canciones asociadas que pueden ser reproducidas.

El usuario selecciona un EM para reproducir, seguidamente el sistema recupera las canciones del EM. Con las canciones recuperadas es generado un fichero playlist que ha de ser descargado en la maquina cliente. Una vez el usuario tiene el fichero, hay que abrirlo con un reproductor de audio (winamp, windows media player, realplayer, ...). El fichero playlist tiene la información necesaria para realizar las peticiones al servidor y reproducir las canciones una a una. (ver anexo 3 para más información sobre playlist).

5.7.1 Ficha de caso de uso

Este caso de uso se ha dividido en dos, ya que hay dos fase claramente diferenciadas: generación del playlist y reproducción del playlist.

CASO DE USO	***Generación playlist***		
Versión	2	**Fecha**	12/05/06
Descripción	El usuario obtiene un playlist con las canciones del EM seleccionado.		
Actores	Usuario		
Precondición	EM seleccionado contiene canciones.		
Flujo principal	1 Recuperar canciones EM 1.1 Escoger EM 1.2 Buscar EM 1.3 Seleccionar las canciones del EM 2 Registrar acción de escuchar EM 3 Crear Playlist 3.1 Por cada canción del EM 3.1.1 Generar información reproducción: título canción, álbum y artista, tiempo archivo, URL peticiones al servidor. 4 Descargar playlist		
Flujos alternativos	Si el EM no contiene canciones se avisara al usuario que no se puede generar el playlist.		
Postcondición	Descargado playlist con las canciones del elemento musical seleccionado.		
Comentarios			

En el paso 2 del caso se observar que se registra la acción de escuchar un EM, esta acción es un requisito de las características para poder generar los top 10 del home.

CASO DE USO	Reproducir PlayList		
Versión	3	**Fecha**	12/05/06
Descripción	Es reproducido en un reproductor de audio el contenido playlist		
Actores	Reproductor		
Precondición	Fichero PlayList normalizado		
Flujo principal	1 Abrir fichero reproductor 1.1 Recuperar información de la canciones 1.2 Por cada canción playlist 1.2.1 Acceder al servidor aplicación (URL canción en playlist) 1.2.2 Buscar usuario valido 1.2.3 Buscar canción valida 1.2.4 Si valido usuario y canción: 1.2.4.1 Configurar conexión reproductor: unidad medida (bytes), tamaño fichero, tipo Mime, información canción 1.2.4.2 Mientras hay datos fichero 1.2.4.2.1 Enviar datos fichero MP3 1.2.5 Registrar canción escuchada por usuario		
Flujos alternativos	Si el usuario o canción son inválidos se aborta la reproducción de la canción. Es registrado en un fichero de logs la causa de la cancelación reproducción.		
Postcondición	Se ha reproducido playlist		
Comentarios			

La responsabilidad de leer los registros del playlist es el reproductor. Este realiza las peticiones de canción una a una, es decir cuando ha acabado de recibir y reproducir los datos de una canción lee el siguiente registro. Esta acción se repite hasta que se hayan realizado las llamadas a todas las entradas del playlist. Es importante destacar como se menciona en el punto 1.2.1 que la URL del playlist tiene toda la información para poder acceder al servidor y recibir los datos de la canción.

5.7.2 Prototipo interface

Ver anexo 3 para visualizar estructura playlist

5.7.3 Diagramas de colaboración simplificados

Para este caso de uso no se ha definido este diagrama ya que con el diagrama de secuencia se visualiza claramente la secuencia de mensajes entre los objetos.

5.7.4 Diagramas de secuencia simplificados

Diagrama de secuencia para el caso de uso *Generación de PlayList:*

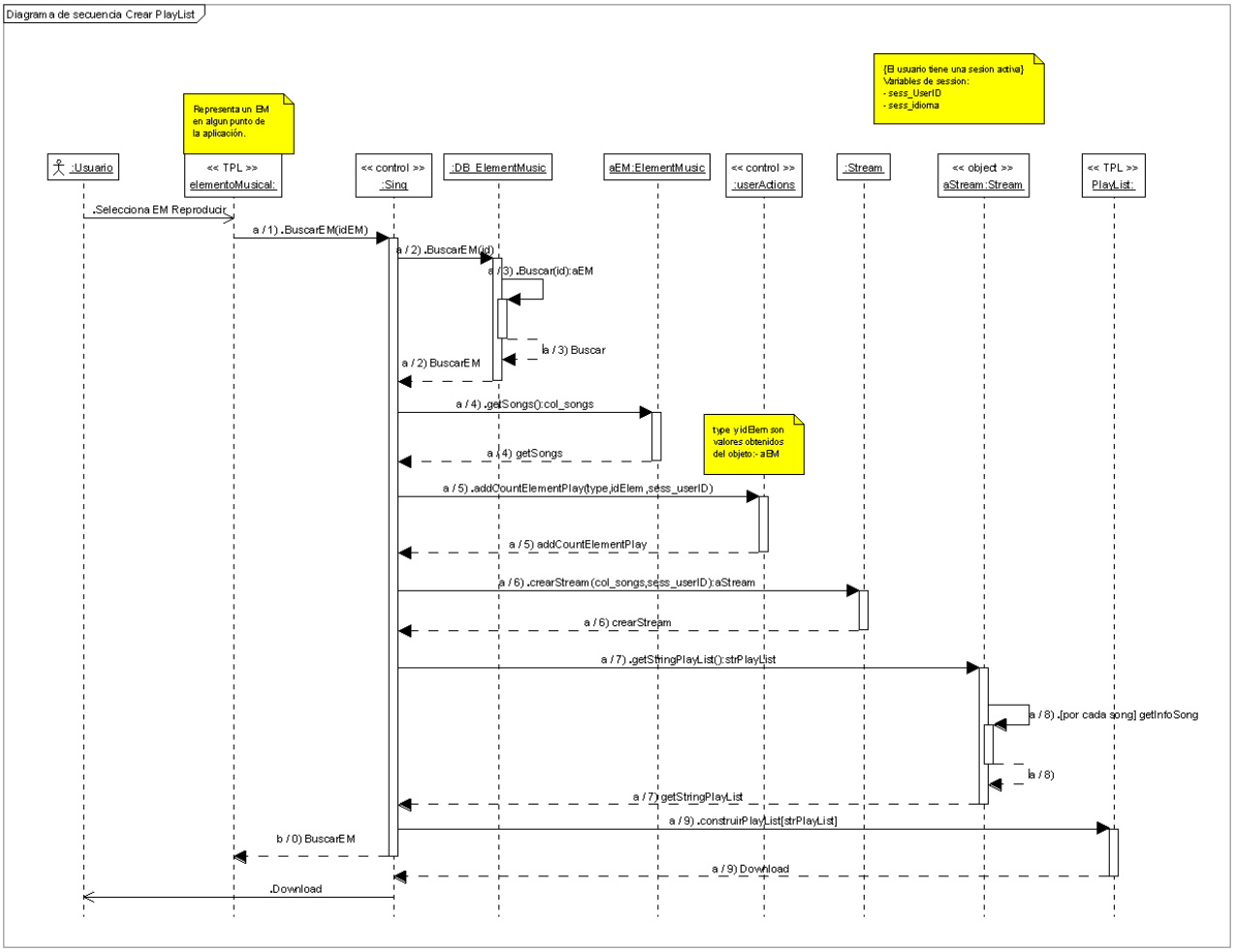


FIG 12.Diagrama de secuencia

Se observar que el controlador *Sing* tiene la responsabilidad de llamar a todas clases para recuperar toda la información antes de generar el playlist.

Como ya se ha dicho en el caso de uso se utiliza el controlador *userActions* para registrar la acción de escuchar, esta información es necesaria para generar los top 10 del home.

Diagrama de secuencia para el caso de uso *Generación de PlayList:*

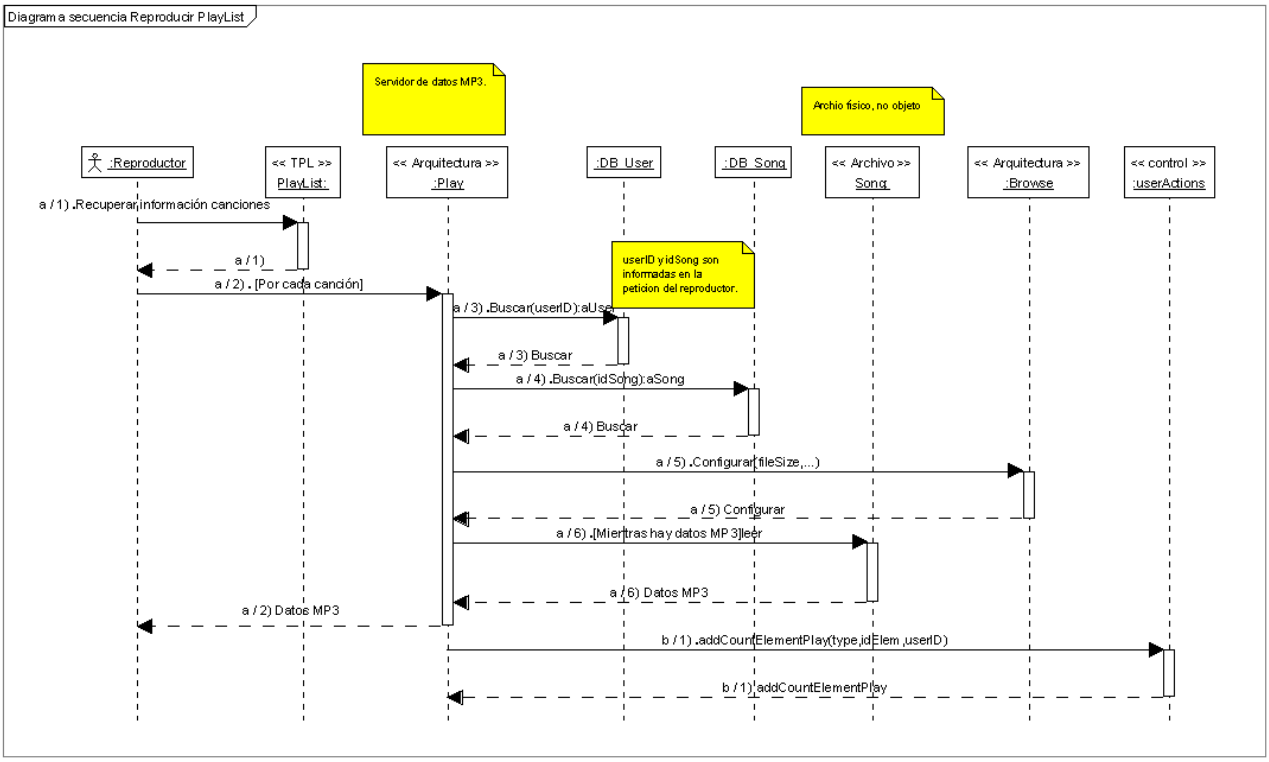


FIG 13: Diagrama de secuencia

En este segundo diagrama hay que destacar que una vez se han enviado todos los datos del fichero MP3 es ejecuta la acción de registrar la reproducción completa (addCountElementPlay).

5.8 Catalogar música

Este caso de uso forma parte del módulo: **Analyce.**

Catalogar música consiste en localizar todos los archivos con formato MP3 que se encuentran en el directorio dado por el administrador y guardar la información necesaria para su posterior localización.

Para llevar acabo la catalogación primeramente se validará el acceso al directorio, a continuación se analizará, incluyendo los subdirectorios. El proceso de análisis consistirá en localizar los ficheros MP3, leer la información musical que contienen y por último guardar en la BD la información obtenida.

5.8.1 Ficha de caso de uso

CASO DE USO	***Catalogar música***		
Versión	3	**Fecha**	13/04/06
Descripción	Catalogar la información de los ficheros MP3		
Actores	Administrador		
Precondición	Cierto		
Flujo principal	1 Entrar directorio base 1.1 Validar directorio 1.2 Mostrar pantalla de inicio 2 Iniciar análisis 2.1 Mientras hay ficheros y directorios 2.1.1 Leer información ficheros 2.1.2 Si es fichero MP3 2.1.2.1 Crear canciones temporales (*tmpSong*) 2.2 Mostrar resultado análisis 3 Catalogar 3.1 Por cada objeto *tmpSong* 3.1.1 Crear Album(idAlbum,AlbumName,Year) 3.1.1.1 Si no existe 3.1.1.1.1 Guardar Album 3.1.2 Crear Genero (idGenere,GenereName) 3.1.2.1 Si no existe 3.1.2.1.1 Guardar Genere 3.1.3 Crear Artista (idArtist,ArtistName) 3.1.3.1 Si no existe 3.1.3.1.1 Guardar Artista 3.1.4 Crear Song (idAlbum, idGenere, idArtist, SongName, Track, UrlFile, EnterDate, Time, FileSize, Bitrate, FileName, Rate, Mode, Mime) 3.1.4.1 Si no existe *3.1.4.1.1 Guardar Song*		

CASO DE USO	Catalogar música
	3.2 Mostrar resultado catalogar.
Flujos alternativos	Si el directorio no es accesible se informa al Administrador y se aborta la operación.
Postcondición	Se ha guardado en la BD la información de los ficheros MP3 del directorio base dado.
Comentarios	

5.8.2 Prototipo interface

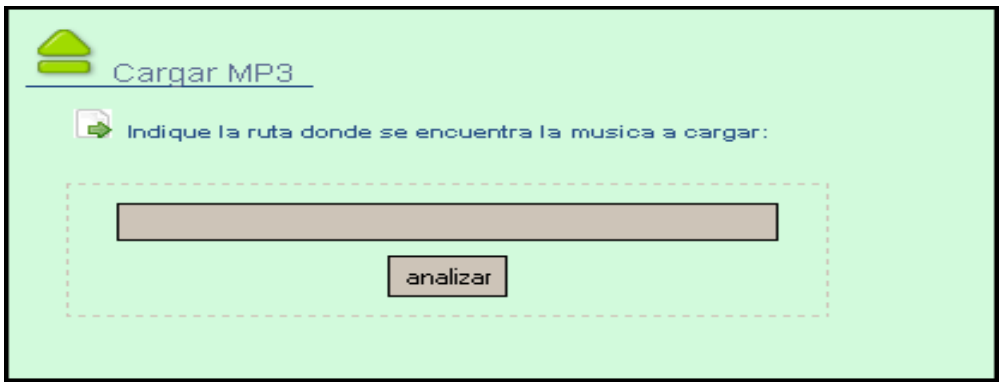


FIG 14: Interface

La FIG 14 muestra la interface a través de la cual el administrador indica el directorio que contiene la música MP3 a catalogar.

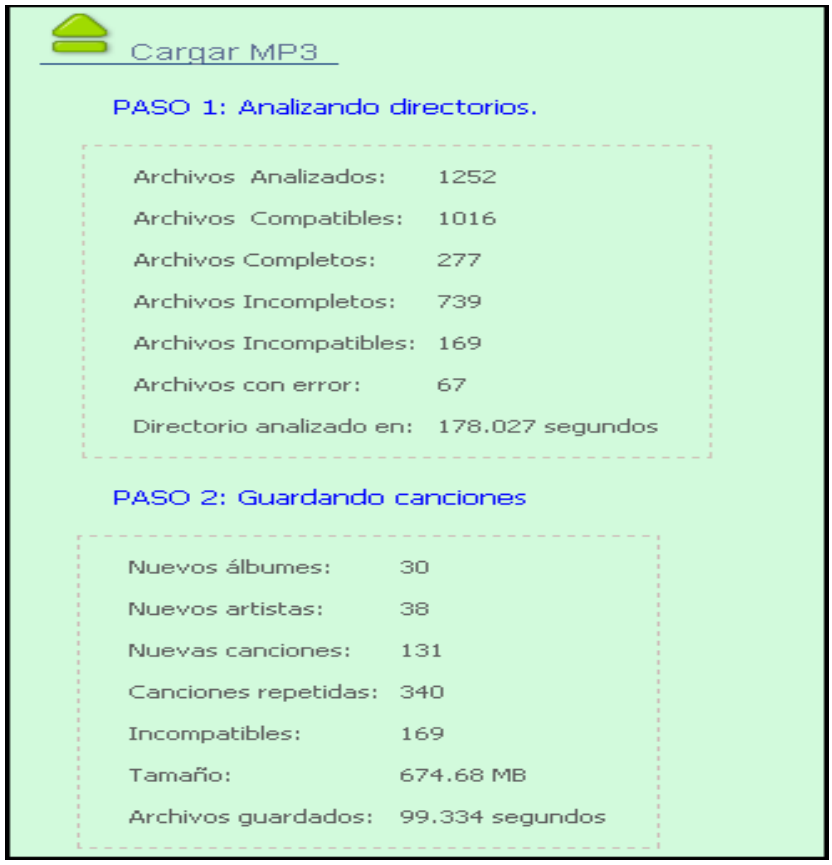


FIG 15: Interface

La FIG 15 muestra la interficies resultante una vez finalizada la carga de archivos.

5.8.3 Diagrama de colaboración simplificado

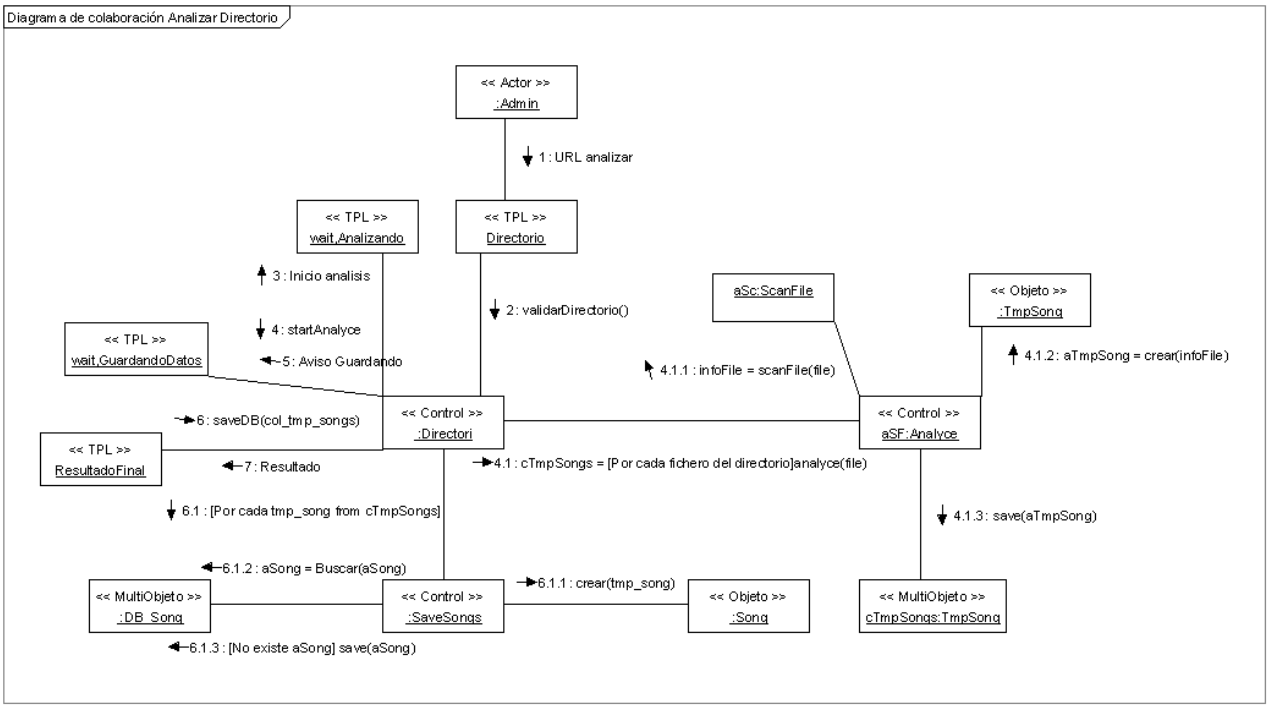


FIG 16: Diagrama de colaboración

El diagrama este compuesto de 3 partes:

La primera, validación de los permisos de acceso al directorio dado por el administrador *(mensaje 2)*.

La segunda parte consiste en analizar todos los ficheros del directorio (*mensaje 4.1*). La clase encargada es "*ScanFile*". Cuando se tiene la información de un fichero y este es MP3 se puede crear una canción temporal (clase *TmpSong, mensaje 4.1.2*). Esta canción es almacenada en una colección de *TmpSongs (mensaje 4.1.3).*

La tercera parte consiste en transformar la información de TmpSongs, en objetos Song, Artist, Album y Genere. Primero se recuperar la información del Artista, Album y Genero. Comprobar su existencia en la BD, si no existen son dados de alta. Una vez se tiene la información actualizada se crea el objeto *Song (mensaje 6.1.1).* Utilizando la información idArtista, idAlbum y el nombre de la canción se comprueba la existencia en la BD (*mensaje 6.1.2*). Si la canción no existe so procede al alta.

En el diagrama no se ha mostrado el proceso de alta para Album,Artista y Genero ya que tiene la misma secuencia de mensajes que el alta del objeto Song.

5.8.4 Nota aclaratoria

Cuando la información obtenida del fichero MP3 es inconsistente, falta el nombre del artista, álbum, genero musical o título de canción. la información quedará distribuida de la siguiente manera:

- Si la inconsistencia es nombre artista la canción formará parte del artista ANONIME.
- Si la inconsistencia es nombre álbum la canción formará parte del álbum ORPHANED.
- Si la inconsistencia es nombre genero la canción formará parte del genero UNKNOWN.
- Si la inconsistencia es título canción el título será el nombre del fichero.

Los elementos ANONIME, ORPHANED y UNKNOWN son dados de alta por la aplicación.

5.9 Buscar información BD pública

Este caso de uso se no se ha implementado, más información en anexos 5 y 6.

5.10 Puntuar elemento musical

Este caso de uso forma parte del módulo: **Browse.**

El módulo B*rowse* permite al usuario navegar por los elementos musicales: álbumes, artistas, géneros y canciones de un álbum.

La puntuación permite definir un grado de satisfacción sobre un elemento musical. Para ello el usuario identifica el EM que desea puntuar, seguidamente selecciona el valor de la puntuación que desea dar. Los valores posibles: entre uno y cinco.

Si es la primera vez que es puntuado el elemento, el valor será sumado al contador global de la aplicación cumpliendo los requisitos de puntuación definido en las característica generales (ver capítulo 3.5.1).

5.10.1 Ficha de caso de uso

CASO DE USO	***Puntuar elemento musical***		
Versión	2	**Fecha**	25/05/06
Descripción	Definir una puntuación sobre un elemento musical		
Actores	Usuario		
Precondición	Cierto		
Flujo principal	1 Puntuar Elemento musical 1.1 Introducir puntuación 1.2 Buscar EM 1.2.1 [Si existe] 1.2.1.1 Modificar puntuación 1.2.2 [No existe] 1.2.2.1 Crear elemento: puntuación, tipo elemento, elemento, id usuario. 1.2.3 [Si puntuación actual es 0] 1.2.3.1 Augmentar el *contador global* (aplicación) con la puntuación nueva. 1.2.4 Guardar cambios		
Flujos alternativos			
Postcondición	Se ha puntuado el elemento seleccionado por el usuario. Si era la primera vez que era puntuado por el usuario se ha actualizado el contador global.		
Comentarios	*Puntuación actual* es la que se encontraba en la BD. Para elementos nuevos o sin puntuar la *puntuación actual* es 0.		

En el caso de uso se observan dos caminos.

El primer camino cuando ya ha sido utilizado por el usuario el elemento musical y por lo tanto esta

dado de alta (1.2.1), solamente hay que modificar la puntuación. Con esta acción se cumple el requisito de puntuar tantas veces como el usuario dese (ver capítulo 3.5.1),

El segundo camino cuando no ha sido utilizado el EM,lo cual hay que dar de alta el uso de este EM con la puntuación, el identificador del EM y el tipo de elemento. El tipo de elemento puede ser: álbum,artista,canción o género musical.

El *contador global* de la aplicación, es un contador compartido por todos los usuarios de la aplicación. Para su incremento se utiliza el valor escogido en la primera votación del usuario. Con esto conseguimos obtener la información para generar el top 10 de las canciones y artistas preferidos (ver capítulo 3.10)

5.10.2 Prototipo interface

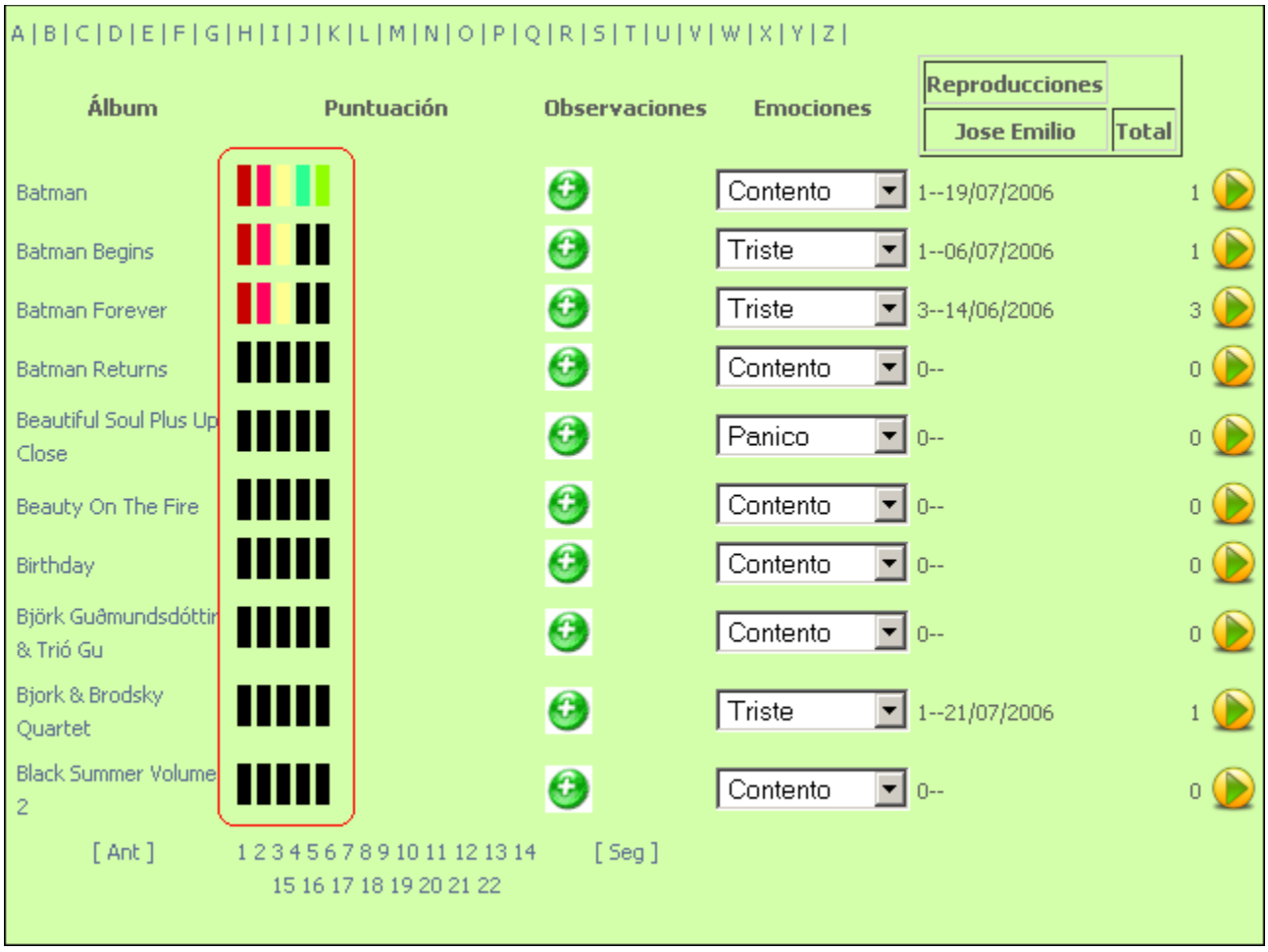


FIG 17: Interface

Los elementos seleccionados de rojo en la FIG 17 proporcionan al usuario la funcionalidad de puntuar, el primer rectángulo negro empezando por la derecha tiene valor uno y el último valor cinco. Al hacer click sobre uno de ellos se realiza la puntuación.

5.10.3 Diagrama de colaboración simplificado

En el diagrama (FIG 18) se observar los dos caminos posibles (2.2) que han sido comentados anteriormente. También podemos ver que una vez se ha asignado la nueva puntuación el usuario se mantiene en el mismo punto del sistema pero con la puntuación del EM actualizada.

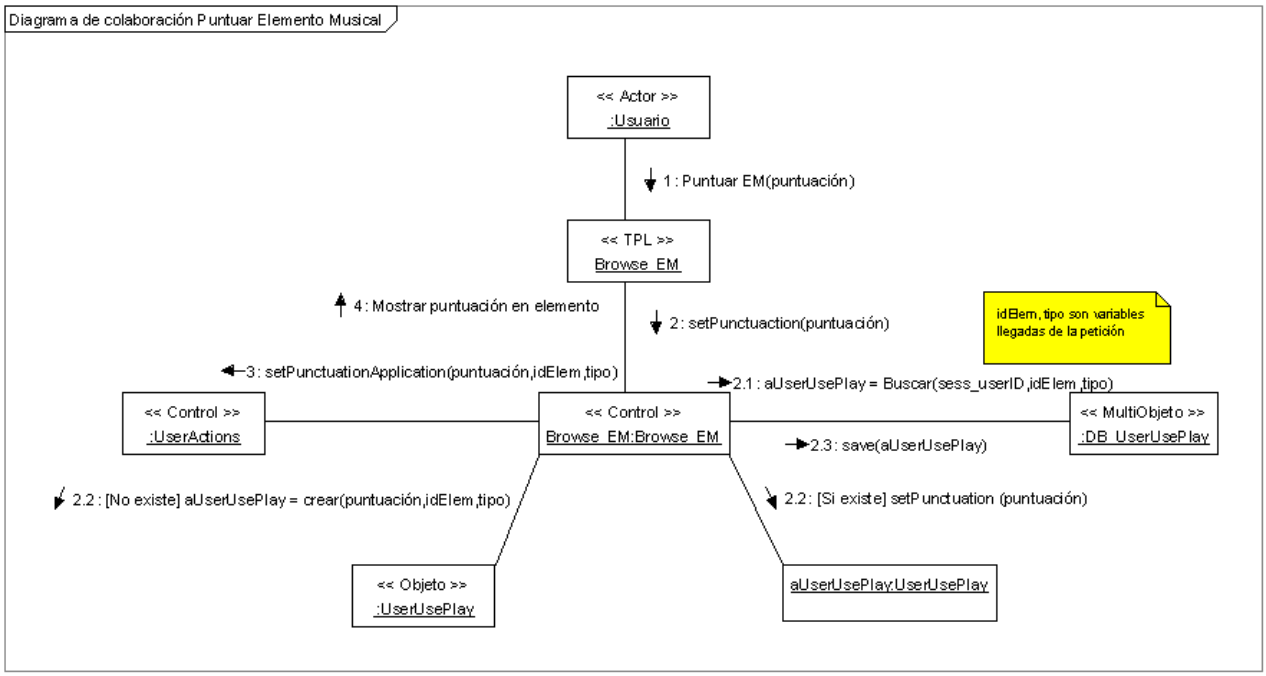


FIG 18: Diagrama de colaboración

5.10.4 Diagrama de secuencia simplificado

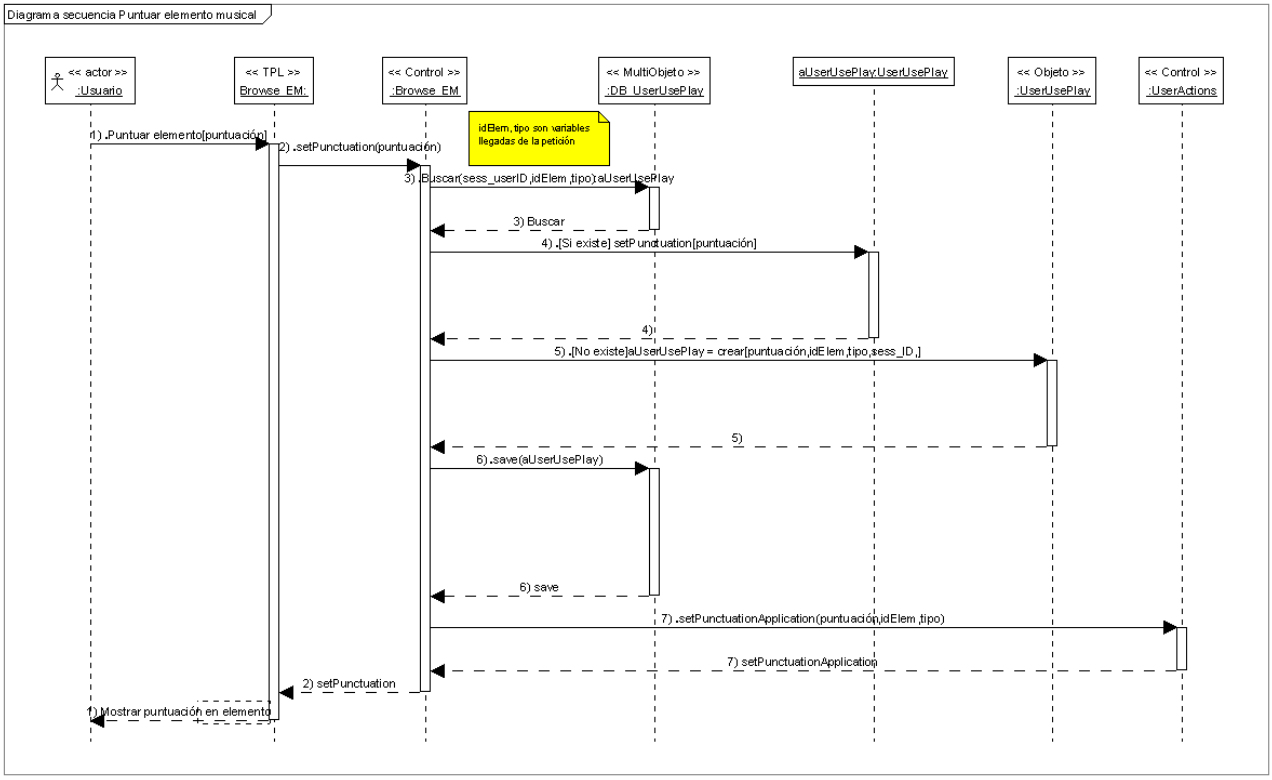


FIG 19.Diagrama de secuencia

5.11 Mantenimiento Observaciones

Este caso de uso forma parte del módulo: **Observaciones.**

El mantenimiento de las las observaciones incluye los casos de uso alta, modificar y eliminar.

En la introducción a los casos de uso se ha especificado que existirán casos que no van a requerir algún tipo de diagrama. Este es el caso, no se van a diseñar los diagramas de colaboración ni secuencia para los procesos de alta y modificación, ya que siguen la misma secuencia de mensajes que los casos de uso: *Registrar usuario* y *Modificar Datos Usuario*. Siguiendo así la metodología Iconix que minimiza el diseño de diagramas innecesarios.

El caso de uso alta y eliminar si que van a ser analizados, mientras que el caso de uso modificar no, ya que no aporta información relevante para el diseño.

5.11.1 Fichas de caso de uso

CASO DE USO	***Alta observaciones***		
Versión	1	**Fecha**	07/06/06
Descripción	Se registran unas observaciones que posteriormente serán asociadas a EM.		
Actores	Usuario		
Precondición	Usuario con sesión activa		
Flujo principal	1 Alta observaciones 1.1 Entrar Datos (Título observación,lugar, amigos, URL, fecha, comentario y una fotografía) 2 Añadir 2.1 Crear 2.2 Añadir observación.		
Flujos alternativos	En caso de no entrar correctamente un dato se muestra un mensaje de error con el campo incorrecto.		
Postcondición	Se ha registrado una observación, el usuario actual es el único propietario.		
Comentarios	El único campo obligatorio es el título de la observación.		

CASO DE USO	Eliminar observaciones		
Versión	3	**Fecha**	07/06/06
Descripción	Eliminar una observación y sus vinculaciones.		
Actores	Usuario		
Precondición	Existe la observación a eliminar.		
Flujo principal	1 Obtener observaciones Usuario 1.1 Recuperar observaciones usuario actual 1.2 Mostrar observaciones 2 Eliminar observación 2.1 Seleccionar una observación 2.2 Eliminar observación de los elementos musicales vinculados 2.3 Eliminar eventos de la agenda que contengan la observación 2.4 Eliminar observación 2.5 Mostrar observaciones		
Flujos alternativos	Si no existe la observación se alerta al usuario.		
Postcondición	Observación y sus relaciones eliminada.		
Comentarios			

Podemos observar en el caso de uso que primero se recuperan todas las observaciones del usuario que tiene la sesión activa. Seguidamente el usuario identificar la observación a eliminar y se activa el proceso de eliminación. Dado que las observaciones están asociadas con EM hay que deshacer esta asociación. Lo mismo ocurre con los eventos de las agenda.

5.11.2 Prototipo interface

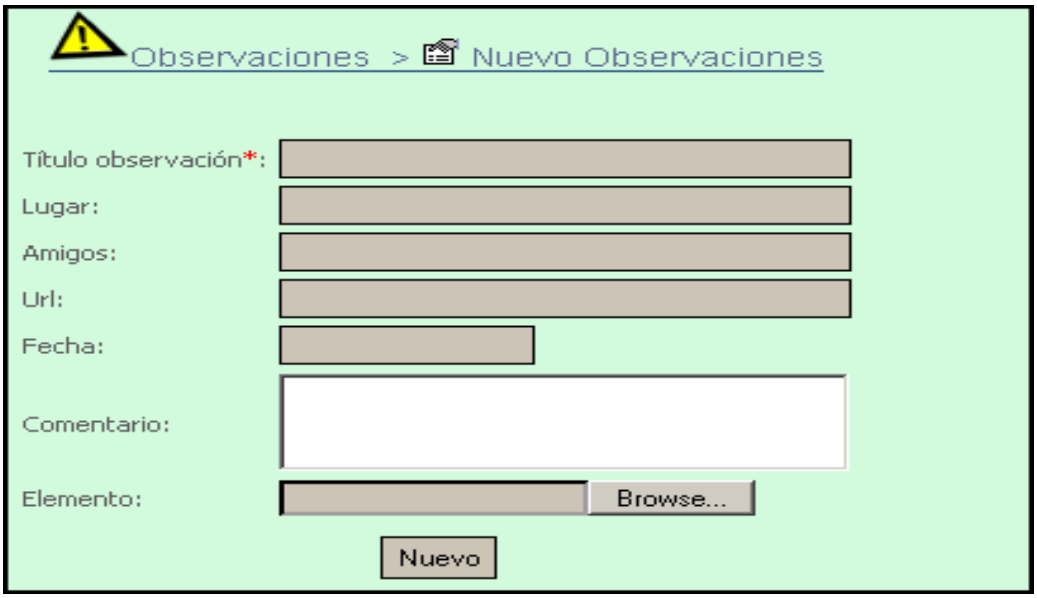


FIG 20: Interface

5.11.3 Diagrama de colaboración

El diagrama de secuencia muestra la información más clara.

5.11.4 Diagramas de secuencia simplificado

Para comprender el caso de uso se han realizado dos diagramas. El primero (FIG 21) informa de la secuencia de mensajes para obtener las observaciones del usuario y el segundo (FIG 22) muestra el proceso de eliminar.

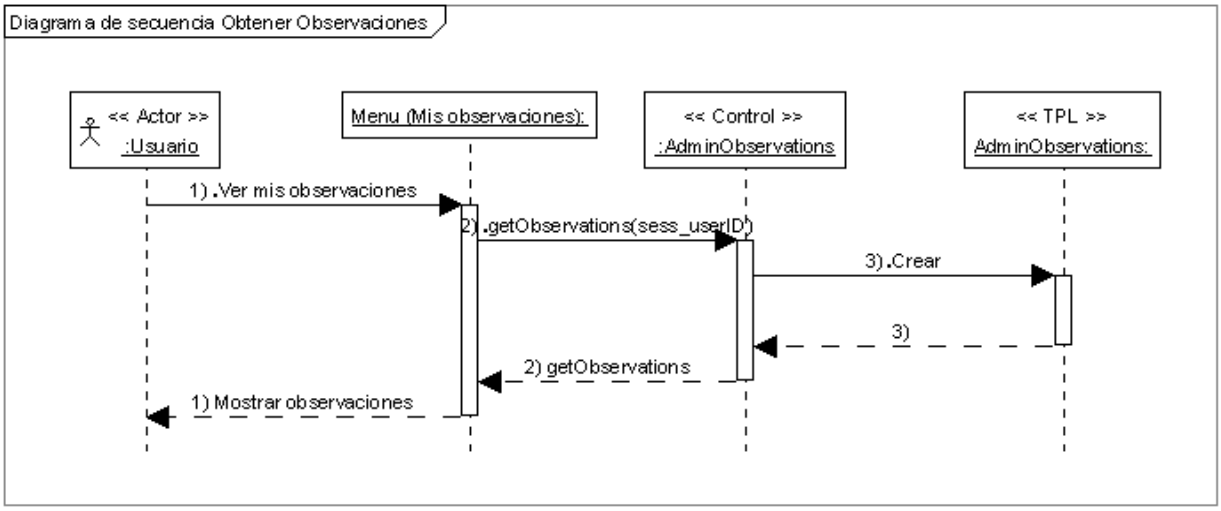


FIG 21: Diagrama de secuencia

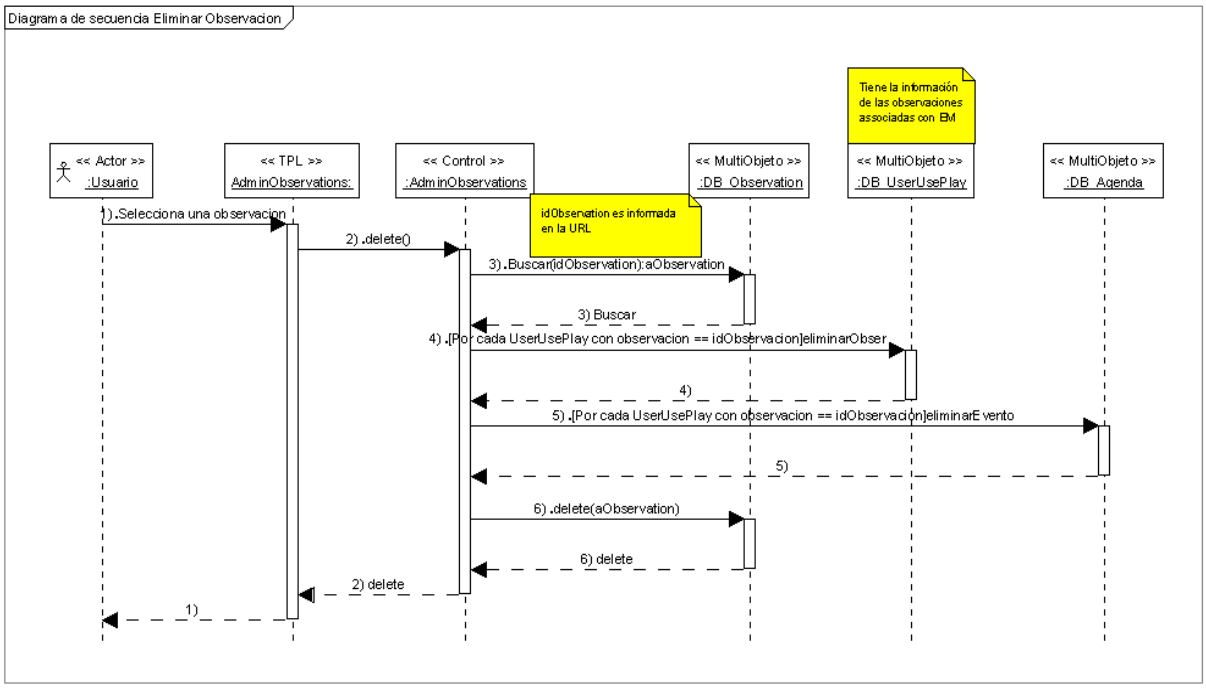


FIG 22: Diagrama de secuencia

La secuencia 4 del diagrama es un proceso iterativo que busca todos los elementos musicales que tiene asociadas a una observación y rompe la asociación . Con la secuencia 5 ocurre lo mismo, un proceso iterativo para buscar todos lo eventos que tiene la observación a eliminar.

Una vez están todos los objetos limpios se elimina la propia observación.

5.12 Asociar Observación

Este caso de uso forma parte del módulo: **Observaciones.**

Como ya se ha comentado en las características generales (capítulo 3.6) las observaciones definidas por un usuario son asociadas a elementos musicales, para poder hacer búsquedas con parámetro personales.

Para realizar una asociación el usuario se ha de situar en la página del sistema donde se muestran un grupo de elemento musicales, seguidamente ha de identificar uno y activar la opción de añadir observación a continuación se le muestran todas las observaciones definidas por él y finalmente selecciona la que desea asociar.

Cuando vuelve a la página donde inicio el proceso de asociación se muestra el EM con la observación.

5.12.1 Ficha de caso de uso

CASO DE USO	***Asociar Observación***		
Versión	2	**Fecha**	08/06/06
Descripción	Asociar una observación de un usuario a un elemento musical		
Actores	Usuario		
Precondición	Observación existente		
Flujo principal	1 Identificar EM y activar proceso asociación 2 Obtener observaciones Usuario 2.1 Recuperar observaciones usuario actual 2.2 Mostrar las observaciones 3 Escoger observación 3.1 Crear asociación 3.2 Mostrar elemento musical con la observación.		
Flujos alternativos			
Postcondición	Asociación de una observación del usuario con EM, puede visualizar-la.		
Comentarios			

El punto 2 del caso de uso es exactamente el mismo que el caso de uso *Eliminar Observaciones* (5.12.1) con la diferencia que el punto de entrada (menú) es un botón al lado del elemento musical, (ver FIG 23). Por ello no se ha realizado el diagrama de secuencia

5.12.2 Prototipo interface

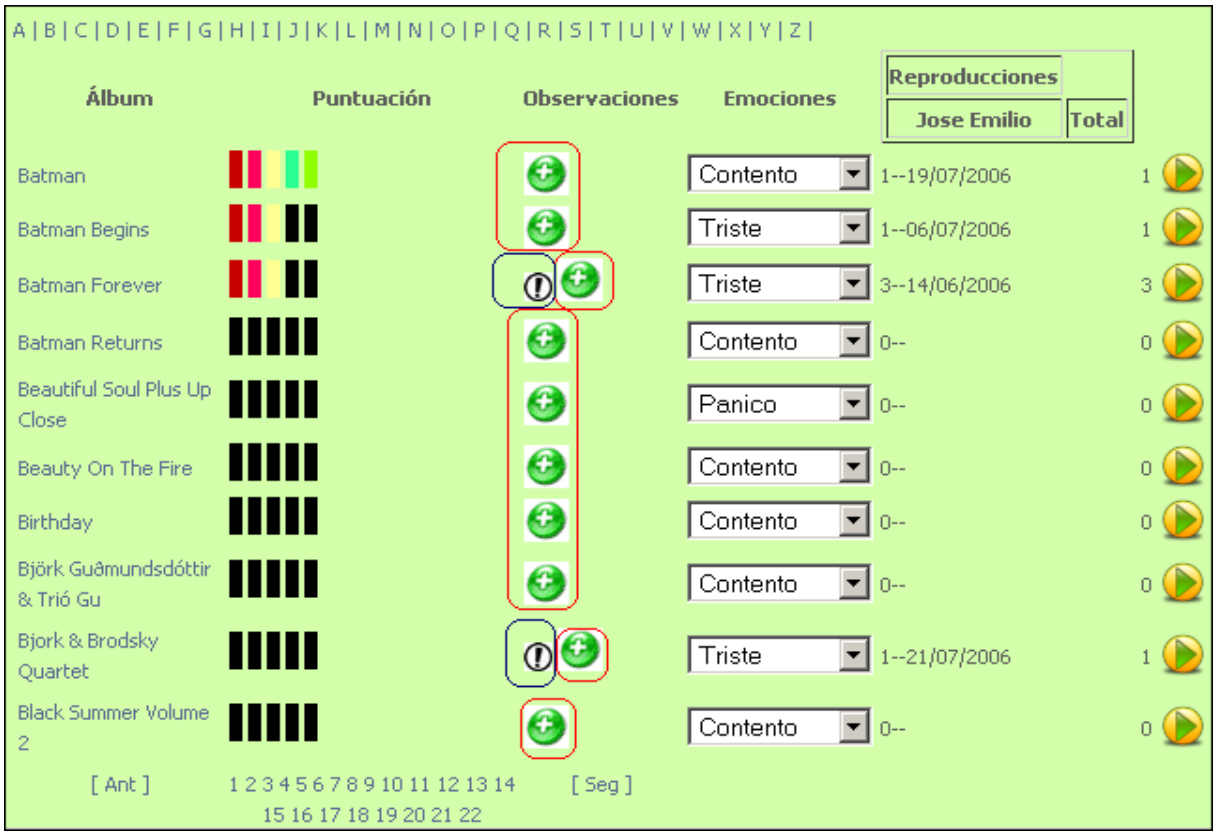


FIG 23.Interface

Los elementos seleccionados en rojo activan el proceso de asociación.

Los elementos seleccionados en azul identifican un elemento con una observación asociada.

La siguiente figura FIG 24 muestra la pantalla para realizar la asociación:

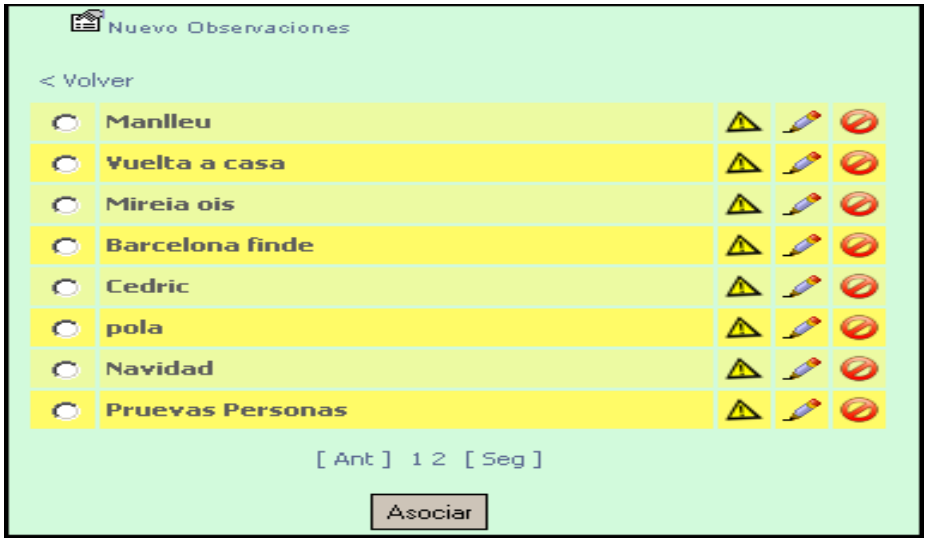


FIG 24: Interface

5.12.3 Diagrama de colaboración simplificado

Como se ha comentado en las aclaraciones del caso de uso, suponemos que el usuario ha activado la opción de asociar, se han recuperado las observaciones del usuario y se dispone a escoger la observación a asociar.

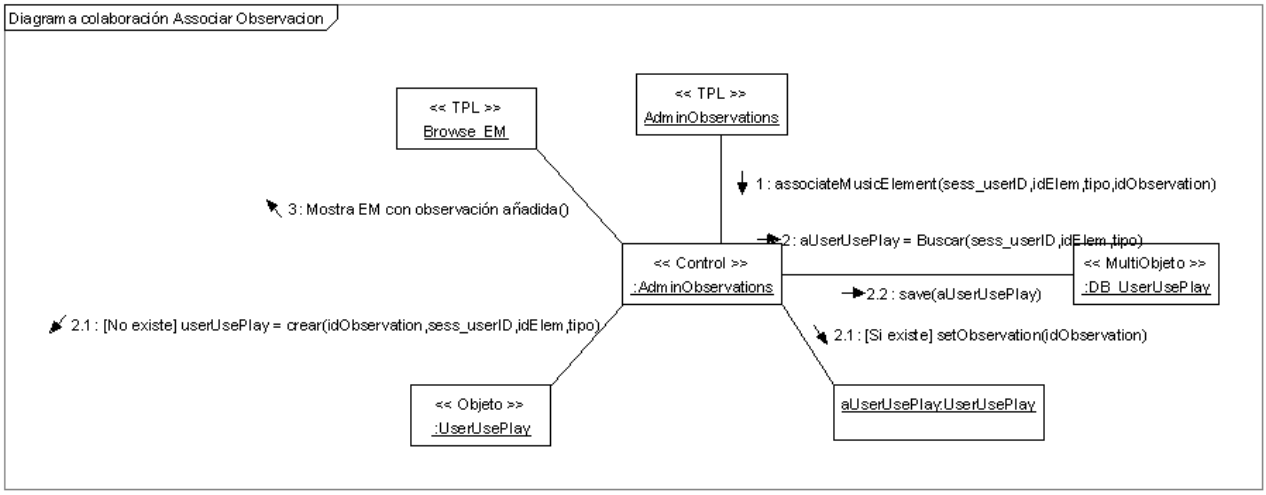
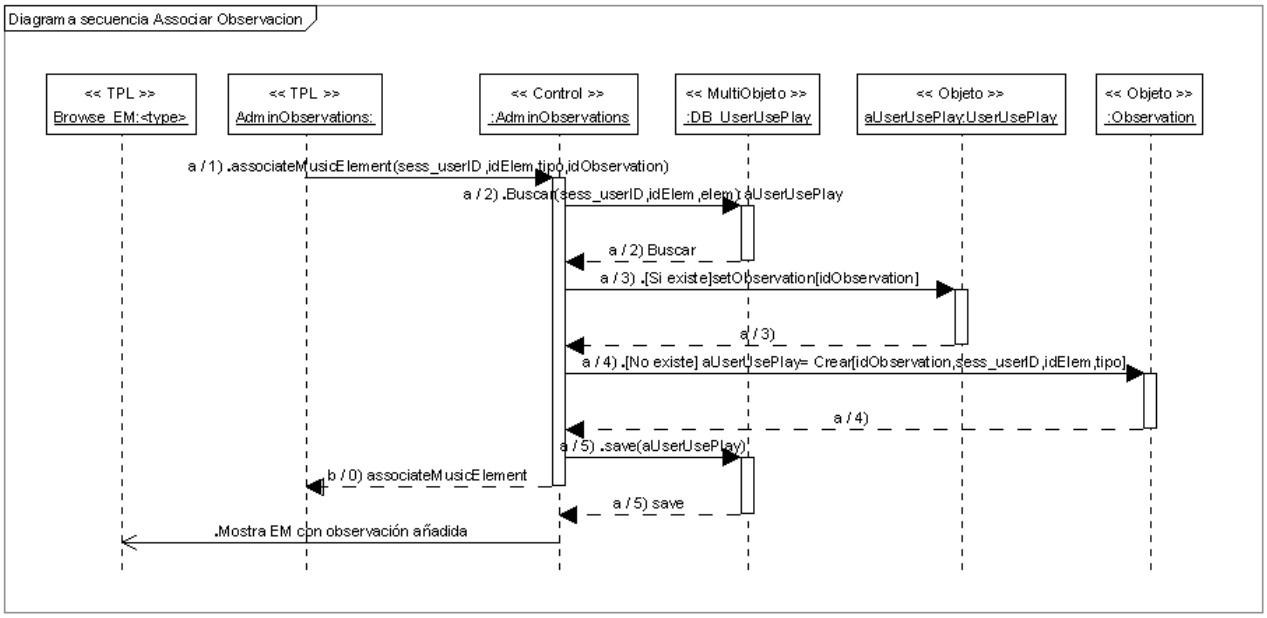


FIG 25.Diagrama de colaboración

En el diagrama hay dos posibilidades para crear la asociación. Uno cuando el usuario ya había realizado alguna operación con el EM y la otra cuando hay que crear el objeto nuevo.

5.12.4 Diagrama de secuencia simplificado



5.13 Mantener Tags

Este caso de uso forma parte del módulo: **Emotions.**

Los tags/emociones son vinculados a EM. Un tag esta compuesto por un nombre que lo identifica y unas preferencias de búsqueda. Al iniciar la aplicación el usuario puede seleccionar un tag y automáticamente se mostrará un resultado con los EM vinculados al tag. Más información en capítulo 3.7.

Los casos de uso del mantenimiento son: *alta. modificar y eliminar*.

De la misma manera que se ha procedido en el casos de uso *Mantenimiento Observaciones* (5.12), solo se van analizar los casos de uso *alta y eliminar*.

Los diagramas de secuencia y colaboración no van a ser necesarios ya que la secuencia de mensajes es similar a los estudiados en el *Mantenimiento Observaciones* (5.12).

5.13.1 Fichas de caso de uso

CASO DE USO	***Alta tags***		
Versión	1	**Fecha**	10/07/06
Descripción	Usuario define un tag personal para poder ser vinculado a EM.		
Actores	Usuario		
Precondición	Usuario con sesión activa		
Flujo principal	1 Alta tag 1.1 Entrar Datos (Nombre identificador, máximo canciones resultantes,EM donde buscar la vinculación, puntuación de los EM, fechas de reproducción). 2 Añadir 2.1 Crear 2.2 Añadir tag.		
Flujos alternativos	En caso de no entrar correctamente un dato se muestra un mensaje de error con el campo incorrecto.		
Postcondición	Se ha registrado un tag, el usuario actual es el único propietario.		
Comentarios	Los campos obligatorios son: nombre identificador y un tipo de EM como mínimo.		

Es importante definir un tipo de EM(artista,canción,álbum o genero) para poder realizar las búsqueda en los tipos de EM seleccionados.

CASO DE USO	Eliminar tag		
Versión	3	**Fecha**	07/06/06
Descripción	Eliminar un tag seleccionado por el usuario actual y sus vinculaciones.		
Actores	Usuario		
Precondición	Existe la observación a eliminar.		
Flujo principal	1 Obtener tags Usuario 1.1 Recuperar tags usuario actual 1.2 Mostrar observaciones 2 Eliminar tag 2.1 Seleccionar un tag 2.2 Eliminar tag de los elementos musicales vinculados 2.3 Eliminar tag 2.4 Mostrar tag		
Flujos alternativos			
Postcondición	El tag elegido y sus vinculaciones han sido eliminadas.		
Comentarios			

5.13.2 Prototipo interface

La FIG 26 muestra la interface que permite dar de alta un tag. El campo con un asterisco rojo es un campo obligatorio.

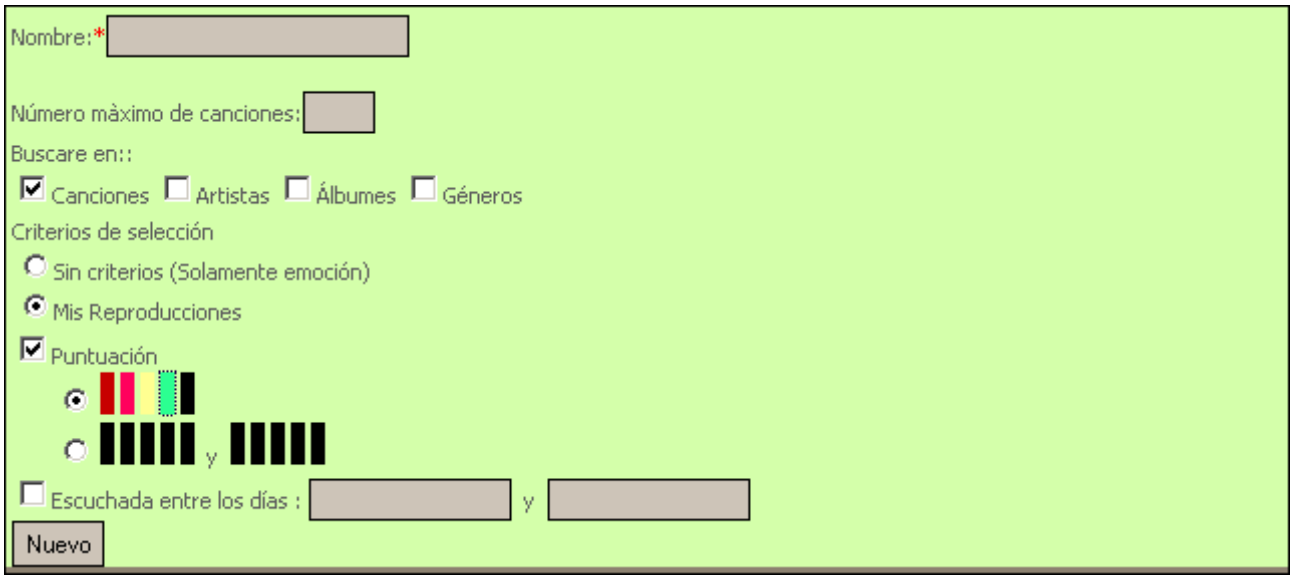


FIG 26: Interface

5.14 Vincular Tag

Este caso de uso forma parte del módulo: **Emotions**.

El usuario identificar el elemento a vincular y escoge la emoción a vincular.

El proceso de vincular un "tag/emoción" es similar a realizar la puntuación o asignar una observación. Por ello solo se va ha especificar el caso de uso. Aquí seguimos aplicando metodología Iconix, no construir diagramas que no aportan información.

5.14.1 Fichas de caso de uso

CASO DE USO	***Vincular Tag***		
Versión	1	**Fecha**	08/06/06
Descripción	Asociar una observación de un usuario a un elemento musical		
Actores	Usuario		
Precondición	Hay Tags de usuario.		
Flujo principal	1 Mostrar interface de navegación. 1.1 Obtener Tags usuario. 2 Vincular Tag 2.1 Seleccionar Tag 2.2 Crear asociación 2.3 Mostrar elemento musical con el Tag		
Flujos alternativos	Si no hay Tags dados de alta, se informar de que no se puede realizar la acción.		
Postcondición	Asociación de un Tag del usuario con EM.		
Comentarios			

Los tags del usuario estarán en la interface (FIG 27) de navegación como el resto de elementos que pueden ser relacionados con EM.

5.14.2 Prototipo interface

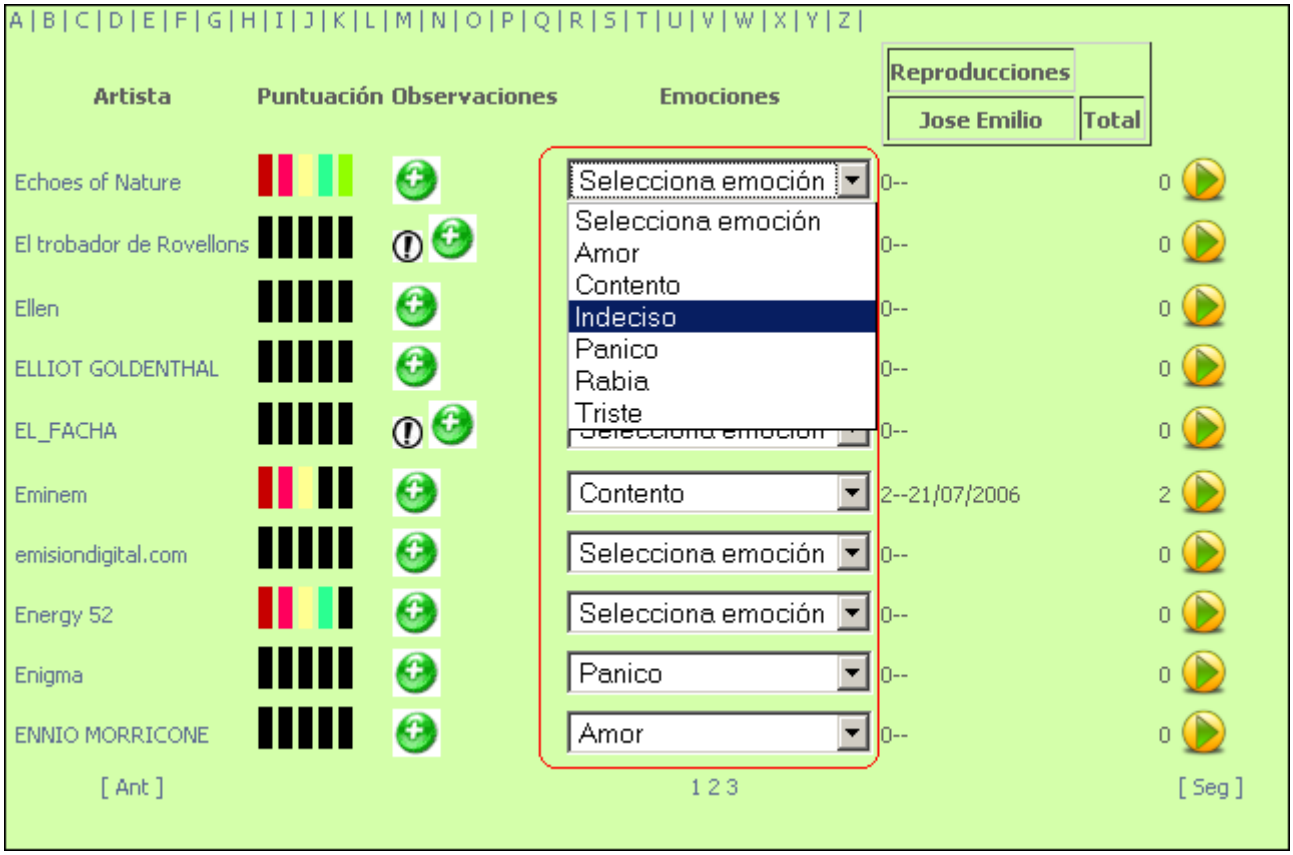


FIG 27. Interface

Los elementos marcados en rojo son los encargados de vincular las emociones. Hay un elemento por cada elemento musical, en este caso por cada artista se puede asociar una emoción.

5.15 Buscar música vinculada Tag

Este caso de uso forma parte del módulo: **Search**

Este caso de uso se activa cuando el usuario se encuentra en las interfaces de la aplicación que permiten seleccionar uno de sus tags. Una vez seleccionado, automáticamente la aplicación buscará todas las vinculaciones echas sobre el tag seleccionado. Las búsqueda sigue los criterios definidos en el tag.

5.15.1 Fichas de caso de uso

CASO DE USO	***Buscar música vinculada Tag***		
Versión	1	**Fecha**	09/06/06
Descripción	Buscar todos EM musicales vinculados a las característica de un Tag.		
Actores	Usuario		
Precondición	Hay Tags de usuario		
Flujo principal	1 Seleccionar un Tag. 1.1 Buscar Tag seleccionado. 1.2 Leer configuración. 2 Buscar música 2.1 Seleccionar todos EM con configuración del Tag seleccionado 2.2 Mostrar Resultado		
Flujos alternativos	Si no hay resultados se informará al usuario.		
Postcondición	Muestran EM vinculado a la configuración del Tag seleccionado		
Comentarios	El resultado de la búsqueda informará que tipo de EM es cada elemento resultante.		

5.15.2 Prototipo interface

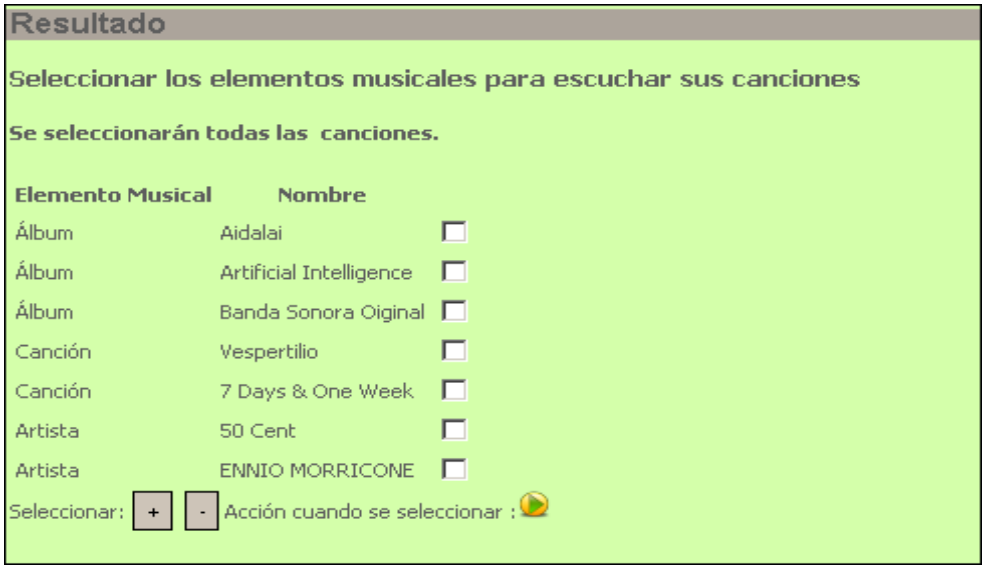


FIG 28. Interface

El resultado de seleccionar un tag posibilita la elección de los EM vinculados.

5.15.3 Diagrama de colaboración simplificado

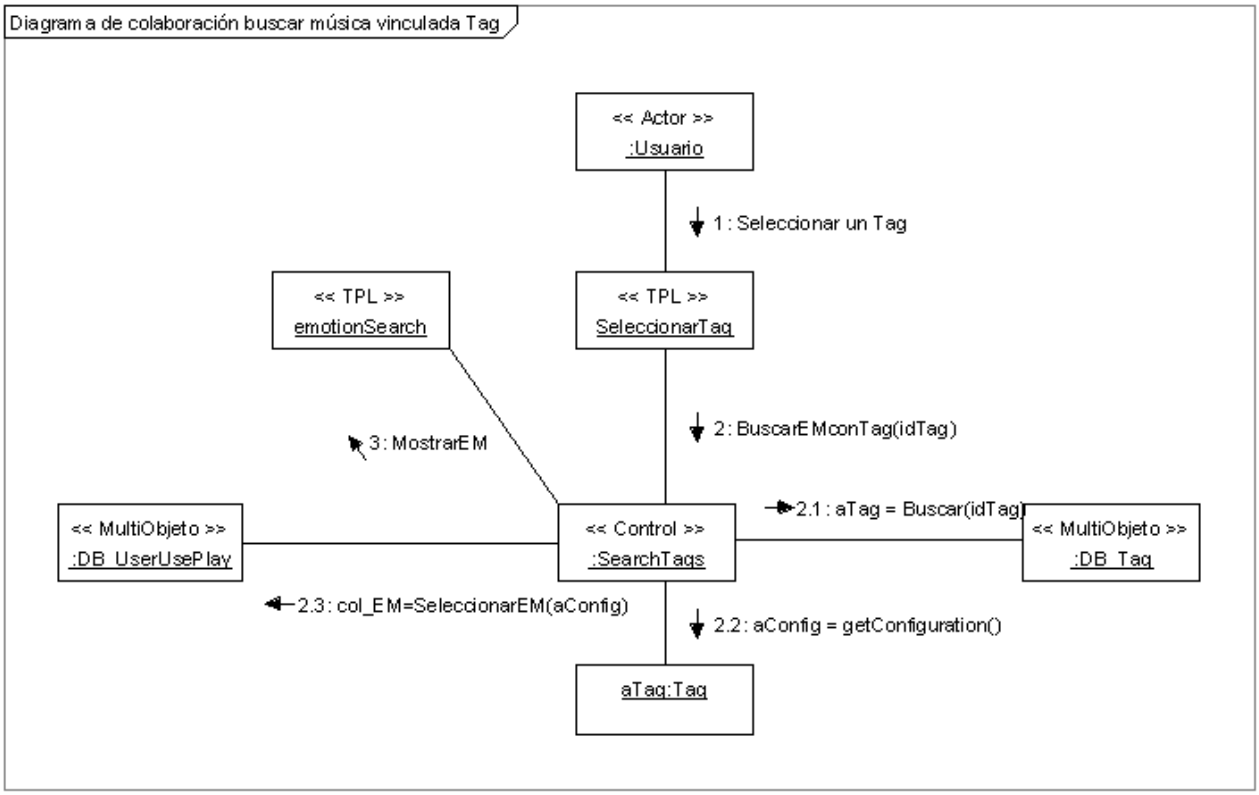


FIG 29. Diagrama de colaboración

El mensaje 2.3 de la FIG 29 buscar todos los EM que cumplen la configuración aConfig. Aconfig

contiene la información que es registrada en el momento del alta del tag (idUsuario,canciones máximas,...). Una vez se han seleccionado se muestran al usuario. En el caso de uso *Reproducir resultados Tag* se detalla como hay se inicia el proceso de reproducción para el usuario.

5.15.4 Diagrama de secuencia simplificado

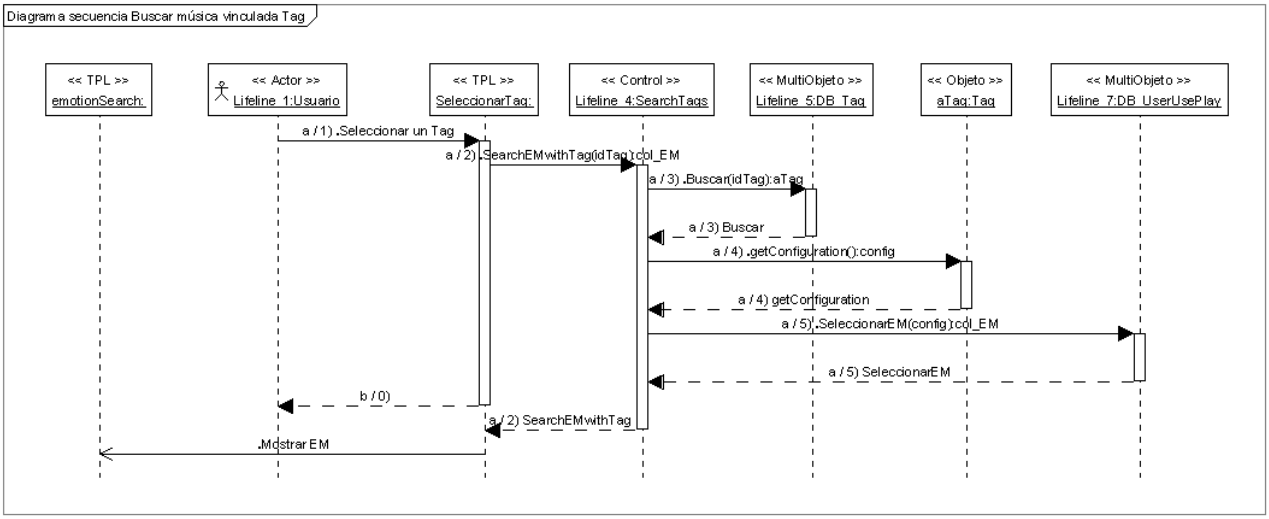


FIG 30: Diagrama de secuencia

5.16 Reproducir resultados Tag

Este caso de uso forma parte del módulo: **Search**

Este caso de uso es la extensión del caso de uso *Buscar Música Vinculada Tag*.

El usuario selecciona los EM musicales que quiere escuchar, una vez realizada la petición la aplicación se encarga de seleccionar todas la canciones de los EM y iniciar el proceso de reproducción, inclusión del caso de uso *Reproducir EM (*ver capítulo *5.7)*

No se ha creído conveniente el diseño de los diagramas de colaboración y secuencia. La interface aporta mayor información de como se realizar el proceso de selección.

5.16.1 Fichas de caso de uso

CASO DE USO	***Reproducir resultado búsqueda Tags***		
Versión	1	**Fecha**	09/06/06
Descripción	Reproduce las canciones de los EM seleccionados por el usuario		
Actores	Usuario		
Precondición	EM con canciones		
Flujo principal	1 Seleccionar EM. 2 Reproducir EM seleccionados		
Flujos alternativos			
Postcondición	Se descarga un playlist con las canciones de los EM seleccionados		
Comentarios			

5.16.2 Prototipo interface

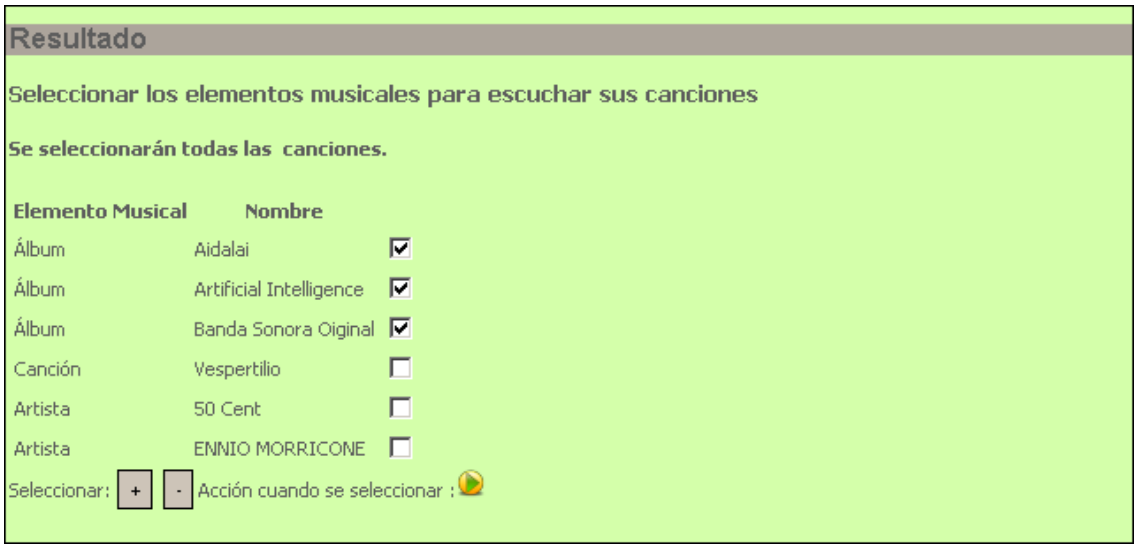


FIG 31: Interface

Se observar que se marcan aquellos elementos a escuchar. Un vez marcados se reproducen utilizando el icono play.

5.17 Mantenimiento PlayList

Este caso de uso forma parte del módulo: **PlayList**.

Un playlist es un conjunto de canciones que puede ser reproducido. Como se ha especificado en las característica de la aplicación (3.8) cada usuario generará sus propios playlist.

Los casos de uso del mantenimiento son: *alta,bajar y modificación.*

Se va ha proceder igual que con el caso de uso *Mantenimiento Tags (5.15),* únicamente se va a detallar el caso de uso alta que es el que aporta mayor información.

5.17.1 Fichas de caso de uso

CASO DE USO	***Alta PlayList***		
Versión	1	**Fecha**	15/07/06
Descripción	Es dado de alta un playlist para poder añadir canciones		
Actores	Usuario		
Precondición	Usuario con sesión activa		
Flujo principal	1 Alta playlist 1.1 Entrar Datos (Nombre identificador) 2 Añadir 2.1 Crear 2.2 Añadir playlist. 3 Añadir canciones		
Flujos alternativos			
Postcondición	Se ha dado de alta un playlist.		
Comentarios	Para añadir canciones ver caso de uso *Añadir canciones*		

5.17.2 Prototipo interface

FIG 32: Interface

5.18 Añadir canciones playlist

Este caso de uso forma parte del módulo: **PlayList**.

Una vez se ha dado de alta el playlist el usuario puede añadir tantas canciones como desee.

Para añadir una canción la aplicación dispondrá de diversos puntos para hacerlo. Por ejemplo en el detalle de un álbum o el resultado de una búsqueda. El usuario identificará la/s canción/es y seleccionará el playlist donde añadirlas.

5.18.1 Fichas de caso de uso

CASO DE USO	***Añadir canciones playlist***		
Versión	1	**Fecha**	15/07/06
Descripción	Son añadidas canciones a un playlist existente		
Actores	Usuario		
Precondición	PlayList dado de alta		
Flujo principal	1 Seleccionar 1.1 Seleccionar canción/es 1.2 Seleccionar playlist usuario 2 Añadir canciones 2.1 Buscar playlist 2.2 Añadir canciones al playlist		
Flujos alternativos	Si no se seleccionar una canción o no se escoge un playlist se advierte al usuario.		
Postcondición	Son añadidas la canciones al playlist seleccionado por el usuario		
Comentarios	Las canciones son añadidas por el final.		

5.18.2 Prototipo interface

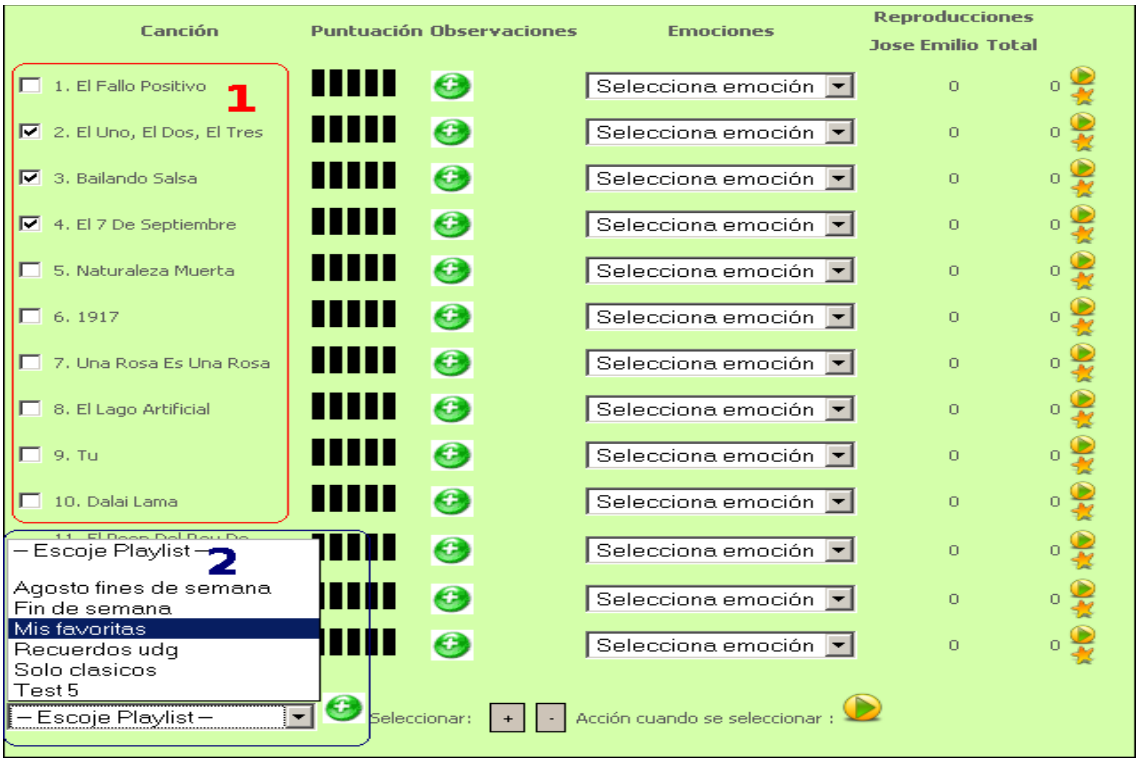


FIG 33.Interface

El número 1 de la FIG 33 son las canciones que van a ser añadidas al playlist. El número 2 son los playlist del usuario.

5.18.3 Diagrama de secuencia simplificado

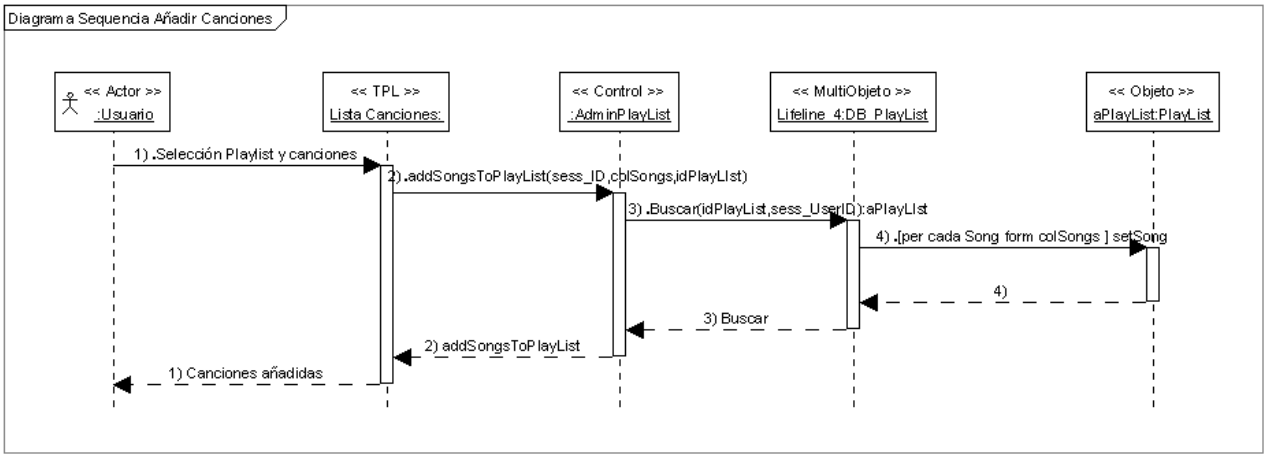


FIG 34.Diagrama de secuencia

El último mensaje del diagrama se encarga de añadir la colección de canciones seleccionada por el usuario, es por ello que se necesita un bucle.

Como ya ocurre en los diagramas presentados anteriormente el identificador de usuario es información proporcionado por la sesión del usuario.

5.19 Buscar Elementos Musicales

Este caso de uso forma parte del módulo: **Search**

El caso tiene la responsabilidad de realizar la búsqueda de elementos musicales (recordamos: artista, álbum canción y genero) coincidentes con una cadena de caracteres entrada por el usuario. El tipo de resultado de la búsqueda será el elemento musical sobre el cual se ha realizado la búsqueda. Ejemplo: Si se ha buscado el nombre de un álbum el resultado serán álbumes coincidentes con el nombre del álbum.

Una vez se tiene el resultado el usuario podrá seleccionar aquellos álbumes que desea reproducir. El siguiente caso de uso se dan mas explicaciones del proceso de reproducción.

5.19.1 Fichas de caso de uso

CASO DE USO	***Buscar Elementos Musicales***		
Versión	1	**Fecha**	23/07/06
Descripción	Busca elementos musicales coincidentes con su sombre.		
Actores	Usuario		
Precondición	Cierto		
Flujo principal	1 Seleccionar 1.1 Seleccionar tipo elemento musical a buscar(album,artista,canción o genero) 1.2 Entrar nombre a buscar 2 Realizar la búsqueda 2.1 Buscar nombre elemento musical 2.2 Mostrar resultado		
Flujos alternativos	Si no se encuentra ningún elemento musical se informar al usuario		
Postcondición	Búsqueda realizada sobre el elemento musical seleccionado.		
Comentarios			

5.19.2 Prototipo interface

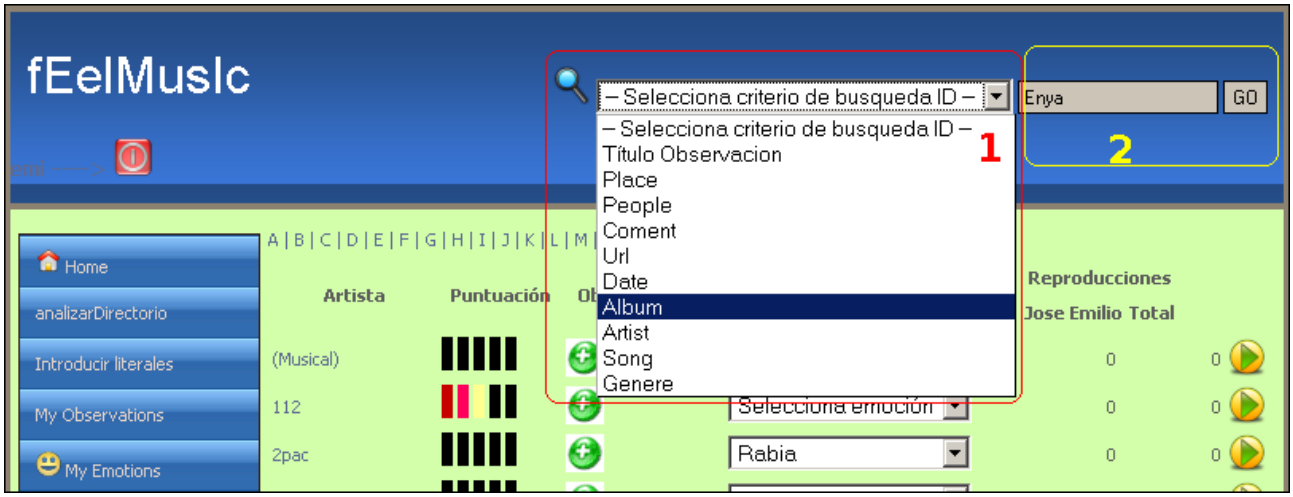


FIG 35.Interface

El número 1 de la FIG 35 identifica los elementos musicales sobre los cuales se puede realizar la búsqueda. La figura número 2 es el lugar donde se introduce el título a buscar.

La figura FIG 35 muestra la búsqueda del elemento musical Álbum con título "Enya".

5.19.3 Diagrama de colaboración simplificado

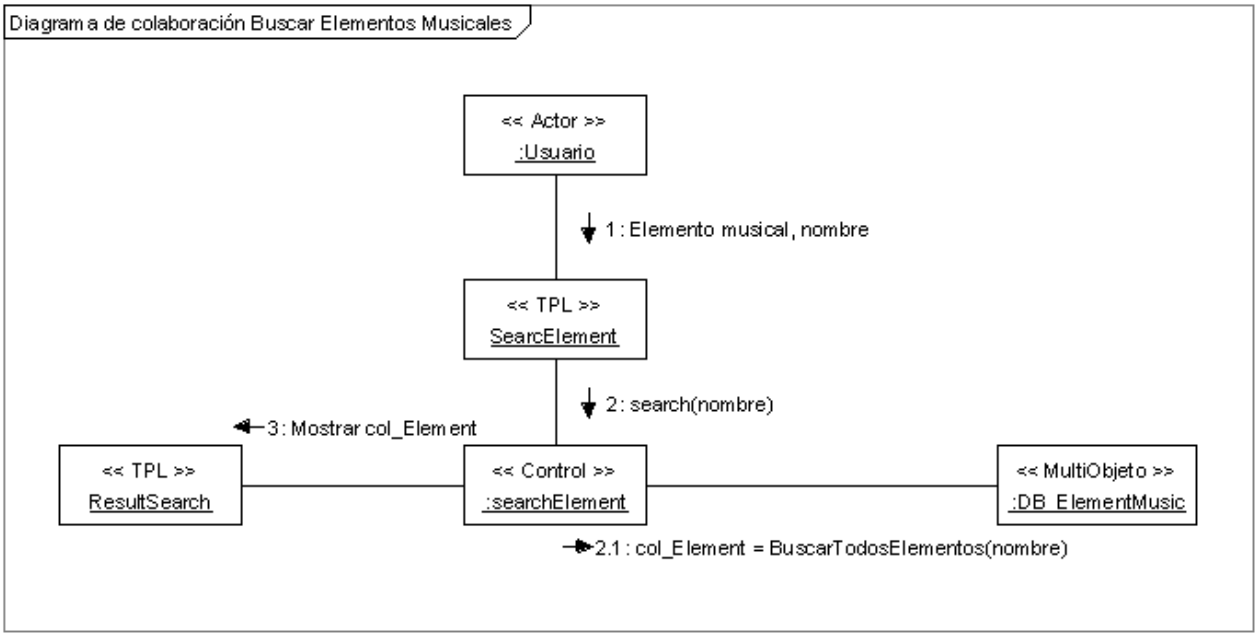


FIG 36.Diagrama de colaboración

El multiobjeto "DB_ElementMusic" es cualquier elemento musical ya sea un álbum,artista,genero o título de canción. Si se tratara de un álbum, la llamada "*buscar todos elementos*" buscará todos los álbumes que tiene en su título la cadena introducida por el usuario. Cuando se han recuperado los elementos son visualizados por pantalla para que el usuario seleccione los que desea escuchar(ver FIG 37 en capítulo 5.20).

5.19.4 Diagrama de secuencia simplificado

No es necesario realizar-lo la información del diagrama de colaboración es suficiente.

5.20 Reproducir resultado elemento musicales

Este caso de uso forma parte del módulo: **Search**

Este caso de uso permite al usuario seleccionar los elementos resultantes de la búsqueda e iniciar la reproducción de las canciones que los componen. Como el caso de uso reproducir ya ha sido tratado en el punto (5.6),siguiendo la metodología Iconix,tal como se ha hecho en otros apartados, simplemente se detallará el caso de uso y la interface.

5.20.1 Ficha de caso de uso

CASO DE USO	***Reproducir Elemento Musical***		
Versión	1	**Fecha**	23/07/06
Descripción	Seleccionar elementos musicales e iniciar su reproducción.		
Actores	Usuario		
Precondición	Se ha realizado una búsqueda previamente, tenemos elementos.		
Flujo principal	1 Seleccionar lista de elementos 2 Reproducir elemento musical		
Flujos alternativos			
Postcondición	Se iniciar el proceso de reproducción con las canciones de los elementos seleccionados.		
Comentarios			

5.20.2 Prototipo interface

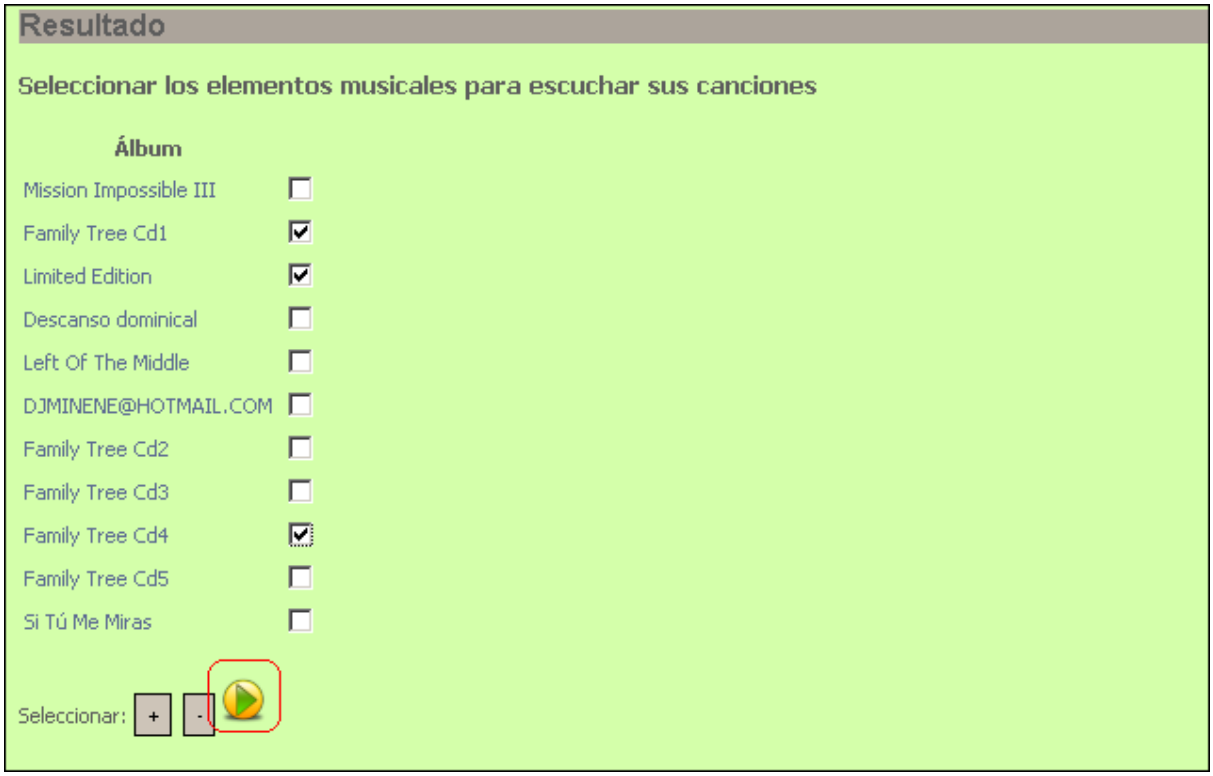


FIG 37.Interface

En la interface FIG 37 se observar los *checkbox* con los cuales el usuario selecciona los elementos musicales a reproducir (álbumes en este caso). El elemento seleccionado en rojo es el encargado de iniciar la reproducción.

5.21 Buscar Observaciones

Este caso de uso forma parte del módulo: **Search.**

Se encarga de buscar las observaciones coincidentes con el campo de búsqueda selecciona por el usuario, ya sea Título observación, Lugar, Amigos, URL, Fecha o Comentario y posteriormente recupera los elementos musicales vinculados en las observaciones coincidentes .

Recuperar los elementos musicales de una observación es tratado con detalle en el caso de uso siguiente (5.22).

5.21.1 Ficha de caso de uso

CASO DE USO	***Buscar Observaciones***		
Versión	1	**Fecha**	23/07/06
Descripción	Buscar observaciones de usuario con una cadena y sus elementos musicales		
Actores	Usuario		
Precondición	Cierto		
Flujo principal	1 Seleccionar campo a buscar de las observaciones 1.1 Seleccionar: Título observación, Lugar, Amigos, URL, Fecha o Comentario. 1.2 Entrar cadena a buscar 2 Buscar observaciones usuario 2.1 Obtener observaciones usuario 2.2 Buscar coincidencia de cadena 3 Buscar elementos musicales de una observación.		
Flujos alternativos			
Postcondición	Muestra los elementos musicales asociados a una observación.		
Comentarios	Para más detalles del punto 4 ver su caso de uso		

5.21.2 Prototipo interface

La interficie se muestra en el siguiente caso de uso.

5.21.3 Diagrama de colaboración simplificado

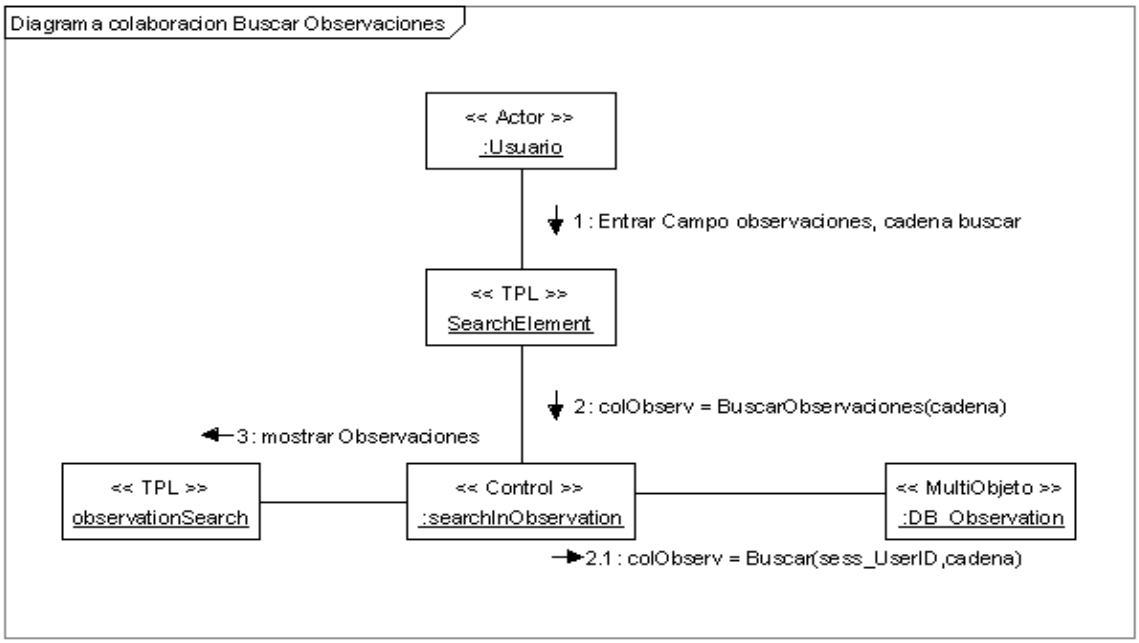


FIG 38.Diagrama de colaboración

El *mensaje 1* del diagrama corresponde a la selección del campo de búsqueda y la cadena a buscar para el campo. El *menaje 2.1* recupera las observaciones del usuario que contienen la cadena a buscar.

Tal como se ha dicho en la introducción del caso de uso el punto 4 de la ficha de caso de uso no es tratado en el diagrama sino en el siguiente caso de uso 5.21.

5.21.4 Diagrama de secuencia simplificado

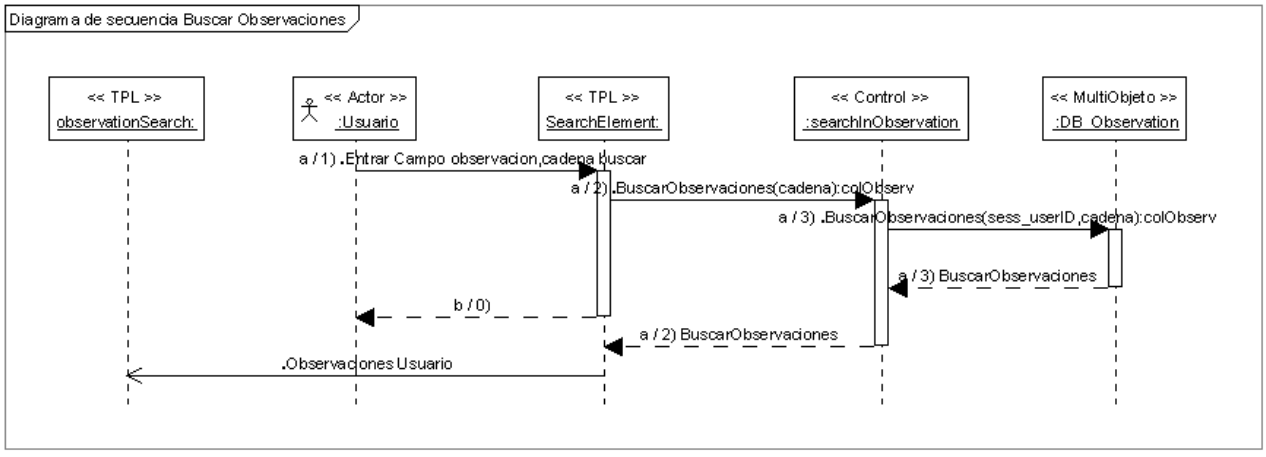


FIG 39.Diagrama de secuencia

5.22 Buscar Elementos Musicales de una observación

Este caso de uso forma parte del módulo: **Search.**

Recupera todos los elementos musicales que han sido relacionados con una observación.

Es utilizado en el caso de uso Buscar Observaciones y ver eventos de la agenda musical.

5.22.1 Ficha de caso de uso

CASO DE USO	***Buscar Elementos Musicales de una observación***		
Versión	1	**Fecha**	30/07/06
Descripción	Buscar elementos musicales asociados a una observación.		
Actores	Usuario		
Precondición	Lista de observaciones del usuario para elegir		
Flujo principal	1 Seleccionar una observación. 1.1 Usuario selecciona una observación de una lista 2 Buscar EM con observación seleccionada 3 Mostrar EM encontrados		
Flujos alternativos	Si no hay ninguna asociación se avisa al usuario		
Postcondición	El usuario obtiene todos los elementos musicales de observación dada de alta por él.		
Comentarios			

5.22.2 Prototipo interface

La interficies es igual que la mostrada en la FIG 28 del capítulo 5.15.

5.22.3 Diagrama de colaboración simplificado

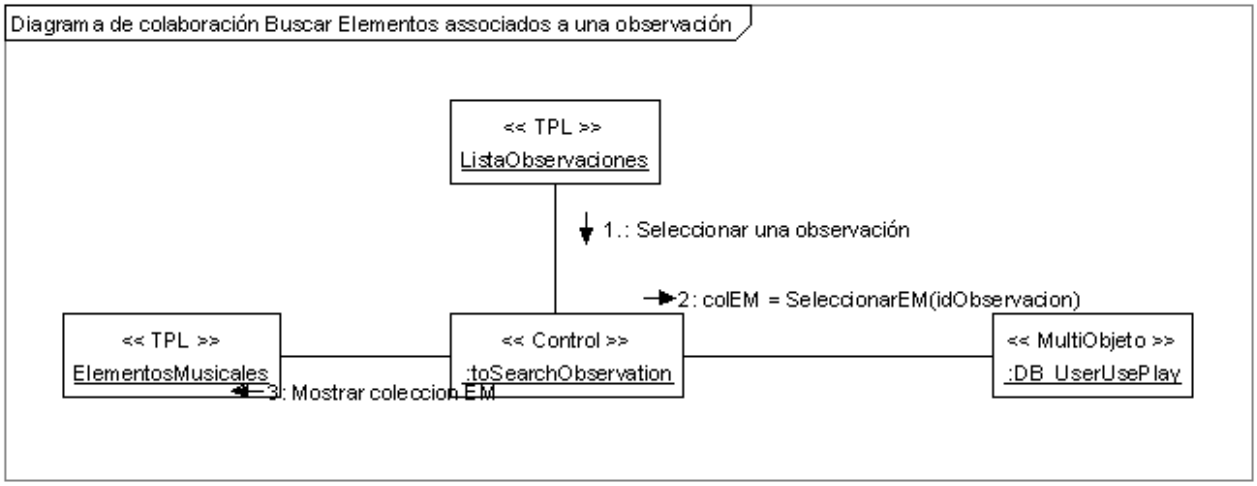


FIG 40: Diagrama de colaboración

La lista de observaciones son el resultado del proceso de búsqueda de observaciones o las observaciones seleccionados en los eventos de la agenda musical.

En el momento que el usuario escoge una de ellas se buscan todas las relaciones en el MuliObjeto *UserUsePlay*, este contiene la información entre usuarios y elementos musicales. Una vez se tienen todos los elementos coincidentes, estos son presentados al usuario,para que pueda escoger cual desea reproducir.

5.23 Estadísticas

Este caso de uso forma parte del módulo: **Home y Stadistics.**

En este apartado se va a tratar un grupo de casos de uso. Se han agrupado ya que todos acceden a los mismos objeto pero consultando diferentes atributos.

Los casos de uso son:

- Mostrar las 10 canciones más escuchadas por usuario
- Mostrar las 10 canciones más escuchadas por total usuarios
- Mostrar las 10 canciones con más puntuación.
- Mostrar los 10 artistas más puntuados
- Mostrar los 10 últimos álbumes añadidos al sistema
- Informar de la última fecha de reproducción de un elemento musical
- Mostrar contador reproducción elemento musical

Se coge como modelo para los diagramas *Mostrar las 10 canciones más escuchadas por usuario*.

5.23.1 Ficha de caso de uso

CASO DE USO	***Mostrar las 10 canciones más escuchadas por usuario.***		
Versión	1	**Fecha**	10/08/06
Descripción	Recupera las 10 canciones mas escuchadas por el usuario actual.		
Actores	Sistema		
Precondición	Cierto		
Flujo principal	1 Recuperar canciones mas escuchadas 1.1 Activar el proceso en el home 1.2 Seleccionar las 10 canciones con máximo contador de reproducción. 2 Mostrar canciones.		
Flujos alternativos	Si no se han reproducido 10 canciones diferentes aparecerán aquellas que se hayan reproducido.		
Postcondición			
Comentarios	Esta estadística aparecerá en el home.		

5.23.2 Prototipo interface

FIG 41: Interface

La interficie muestra los top 10: canciones más escuchadas entre todos los usuarios de la aplicación y las canciones más escuchadas por el usuario actual.

5.23.3 Diagrama de colaboración simplificado

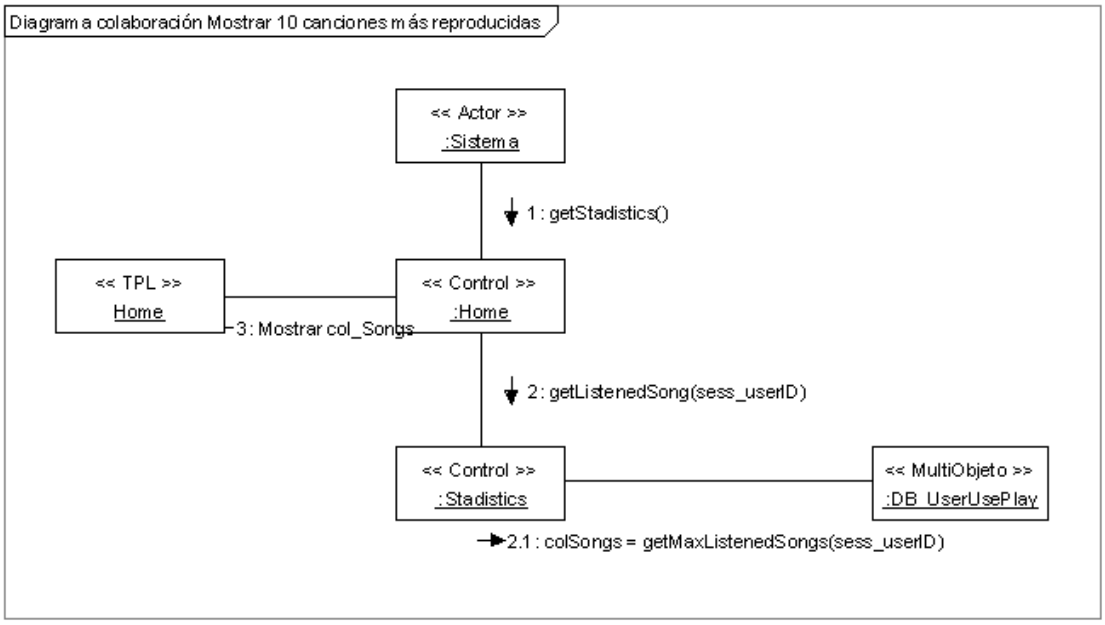


FIG 42.Diagrama de colaboración

En el diagrama de la figura observa que el proceso es activado sobre el controlador home (*mensaje 1*). Este es el conocedor del controlador estadísticas que se encarga de recuperar las 10 canciones más escuchas del usuario con la sesión activa (*Mensaje 2.1*). Por último se presentan el resultado en la pagina home. (*Mensaje 3*).

5.23.4 Nota Diseño

La información que identificar a todos los usuarios será miembro de un usuario interno de la aplicación. Este será dado de alta al instalar la aplicación con identificador 1.

Del resto de casos de uso hay que tener en cuenta que se necesitarán los siguientes atributos en el objeto userUsePlay: Contador reproducción, Contador de puntuación, fecha última reproducción y fecha de alta en el sistema.

5.24 Mantenimiento Eventos Agenda musical

Este caso de uso forma parte del módulo: **Agenda.**

El mantenimiento esta compuesto por 3 casos de uso: *Alta,Baja y Eliminar.* Tal como se ha realizado en anteriores casos de uso (Mantenimiento Observaciones,Mantenimiento Tags y Mantenimiento Playlist) solo se va a detallar el caso de uso Alta ya que es el que aporta mayor información. Los diagramas de colaboración y secuencia no son necesarios. Una vez más se aplica las técnicas de la metodología Iconix.

5.24.1 Ficha de caso de uso

CASO DE USO	***Alta Evento Agenda***		
Versión	1	**Fecha**	07/08/06
Descripción	Dar de alta evento personal para un usuario		
Actores	Usuario		
Precondición	Usuario con sesión activa		
Flujo principal	1 Alta evento agenda 1.1 Entrar Datos (Fecha aviso, descripción y Observación personal a recordar) 2 Añadir 2.1 Crear 2.2 Añadir evento de la agenda musical		
Flujos alternativos	Si la fecha de aviso no es superior al día actual del sistema, avisará del error.		
Postcondición	Se ha dado de alta un evento.		
Comentarios	Los 3 campos son obligatorios.		

5.24.2 Prototipo interface

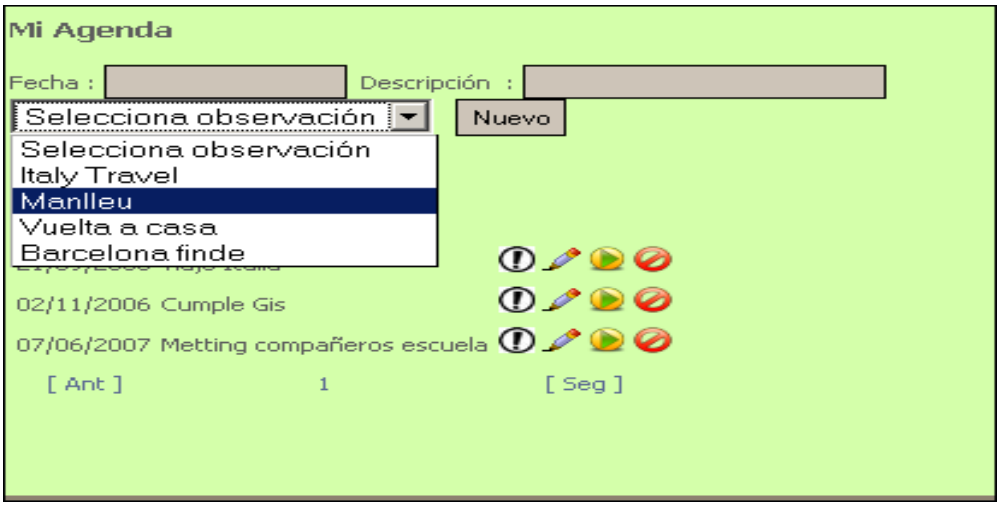


FIG 43: Interface

En la interface se muestra los campos donde el usuario entrará los datos. La fecha, la descripción del evento y la observación a recordar.

5.25 Ver próximos eventos

Este caso de uso forma parte del módulo: **Agenda.**

Selecciona los eventos del usuario que ha iniciado una sesión. Los eventos mostrados serán aquellos que su fecha sea igual o superior a la fecha actual del sistema y estarán ordenados por fecha creciente.

Los eventos del día actual serán marcados con otro color.

5.25.1 Ficha de caso de uso

CASO DE USO	***Ver próximos eventos***		
Versión	1	**Fecha**	07/08/06
Descripción	Visualiza los eventos de la agenda del usuario para los días con fecha superior al día actual del sistema.		
Actores	Usuario		
Precondición	Usuario con sesión activa		
Flujo principal	1 Seleccionar eventos 1.1 Seleccionar los eventos del usuario actual de sistema con fecha superior a la actual. 2 Mostrar eventos		
Flujos alternativos	Si no hay eventos se muestra un mensaje informativo.		
Postcondición	Se muestran los eventos del usuario con fecha superior a la actual		
Comentarios	Los eventos del día actual son marcados de otro color.		

5.25.2 Prototipo interface

FIG 44. Interface

Tal como se ha descrito en la introducción del caso de uso, la interface muestra todos los eventos superiores a la fecha actual del sistema (04/09/2006) y con un color mas llamativo aquellos que corresponden al día actual del sistema.

5.25.3 Diagrama de colaboración simplificado

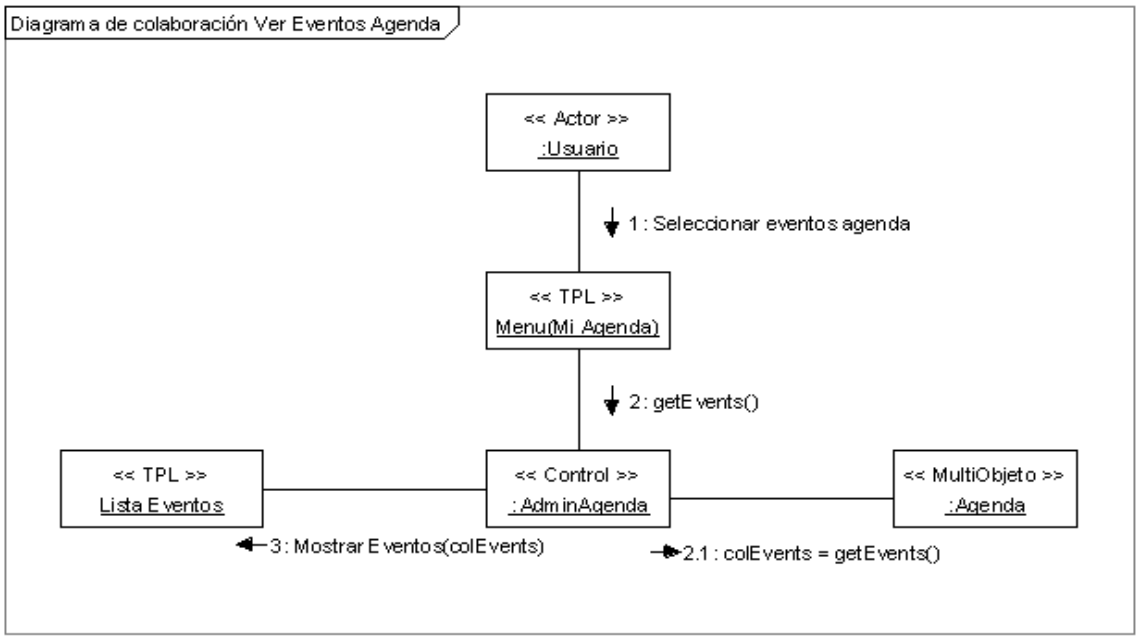


FIG 45. Diagrama de colaboración

En el diagrama se observa como el usuario activa el caso de uso mediante el menú de navegación .Seguidamente la clase de control *AdminAgenda* realiza la petición sobre el MultiObjeto *Agenda* el cual de vuelve todos los Eventos que tiene que ser mostrados.

5.25.4 Diagrama de secuencia simplificado

No es necesario ya que el diagrama de colaboración aporta toda la información necesaria.

5.26 Avisar eventos

Este caso de uso forma parte del módulo: **Agenda.**

Este caso tiene la responsabilidad de buscar los eventos que el usuario programó para el día actual del sistema.

5.26.1 Ficha de caso de uso

CASO DE USO	***Avisar Eventos***		
Versión	1	**Fecha**	07/08/06
Descripción	Avisa de manera automática de los eventos de la agenda para la fecha actual		
Actores	Sistema		
Precondición	Usuario con sesión activa		
Flujo principal	1 Buscar eventos 1.1 Buscar si existe como mínimo un evento para la fecha actual del sistema. 2 Mostrar mensaje informativo.		
Flujos alternativos	Si no hay ningún evento no se muestra ningún mensaje.		
Postcondición	Se muestra un mensaje para informar de que hay eventos para la fecha actual del sistema.		
Comentarios	El mensaje aparece en una parte de la aplicación dónde hay información varia.		

5.26.2 Prototipo interface

FIG 46: Interface

El elemento seleccionado en rojo es el encargado de avisar que hay eventos para el día actual.

5.26.3 Diagrama de colaboración simplificado

No se ha creído necesario el diseño ya que los objeto que intervienen y la secuencia de mensajes es la misma que el caso de uso anterior, *eventos Agenda.* La diferencia se encuentra sobre la interface de salida, en este caso se trata de un mensaje y no de los eventos.

5.26.4 Diagrama de secuencia simplificado

No es necesario tal como se ha especificado en el apartado anterior.

5.27 Reproducir música del evento

Este caso de uso forma parte del módulo: **Agenda.**

Como los eventos de la agenda musical tiene una observación, la funcionalidad de este caso de uso es recuperar los elementos musicales asociados a la observación. Seguidamente el usuario selecciona aquellos elementos que quiere escuchar e inicia el proceso de reproducción.

5.27.1 Ficha de caso de uso

CASO DE USO	***Avisar Eventos***		
Versión	1	**Fecha**	07/08/06
Descripción	Reproducir los elementos asociados a una observación.		
Actores	Usuario		
Precondición	Usuario con sesión activa		
Flujo principal	1 Seleccionar evento 2 Buscar Elementos asociados a observación 3 Reproducir Elemento Musical.		
Flujos alternativos			
Postcondición	Reproducir elementos musicales seleccionados por el usuario.		
Comentarios	Los puntos 2 y 3 son casos de uso ya tratados.		

Tal como se especifica en la ficha de caso de uso los puntos dos y tres corresponde a los casos de uso *Buscar Elementos asociados a observación (*ver capítulo 5.22*) y Reproducir Elemento Musical (*ver capítulo 5.7*)*

5.28 Mantenimiento Canciones Favoritas

Este caso de uso forma parte del módulo: **FavoriteSongs.**

Permite al usuario realizar accesos directos sobre sus canciones favoritas.

Los casos de uso que forman el mantenimiento son: Alta y Eliminar. Como se ha procedido con los mantenimientos anteriores únicamente se detalla el caso de uso alta. El resto de diagramas no aportan ningún tipo de información relevante.

5.28.1 Ficha de caso de uso

CASO DE USO	***Alta***		
Versión	1	**Fecha**	07/08/06
Descripción	Crea un acceso directo a la canción seleccionada por el usuario.		
Actores	Usuario		
Precondición	Usuario con sesión activa		
Flujo principal	1 Seleccionar canción 2 Añadir 2.1 Crear 2.2 Añadir enlace.		
Flujos alternativos			
Postcondición	La canción seleccionada por el usuario es dada de alta como canción favorita.		
Comentarios			

5.28.2 Prototipo interface

FIG 47. Interface

La interface de la FIG 47 muestra las canciones que han sido declaradas por el usuario como favoritas.

5.29 Buscar páginas web sobre un artista.

Este caso de uso forma parte del módulo: **Links.**

Cuando el usuario necesita buscar páginas web con información de un artista, puede utilizar esta funcionalidad.

5.29.1 Ficha de caso de uso

CASO DE USO	***Buscar paginas web sobre artista***		
Versión	1	**Fecha**	30/08/06
Descripción	Buscar páginas web con información de artistas.		
Actores	Usuario		
Precondición	Usuario con sesión activa		
Flujo principal	1 Introducir o escoger nombre artista. 2 Buscar paginas web artista en http://api.google.com 3 Mostrar resultados		
Flujos alternativos	Si falla la conexión con google se avisa al usuario.		
Postcondición	Muestra los diez primeros resultados de la búsqueda en http://api.google.com		
Comentarios	Utilizar técnicas SOAP.		

Tal como se comenta en el comentario del caso de uso la técnica utilizada para obtener la información del artista es SOAP (ver capítulo 6.7).

5.29.2 Prototipo interface

FIG 48: Interface

El elemento seleccionado en rojo permite realizar las búsquedas de links en google.

5.29.3 Diagrama de colaboración simplificado

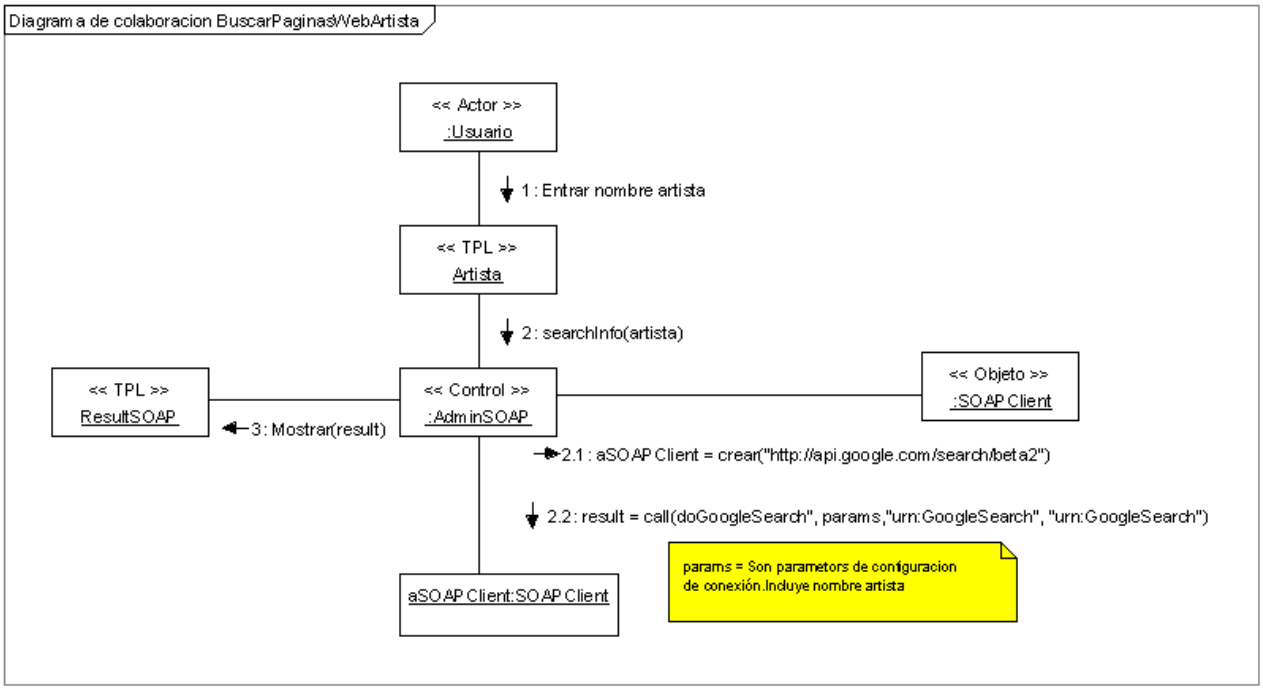


FIG 49.Diagrama de colaboración

Como se observa en el diagrama hay que crear un objeto *SoapClient (mensaje 2.1)* que se encarga de realizar la búsqueda en google(*mensaje 2..2*). Una vez se tiene el resultado se muestra por pantalla (*mensaje 3*)

5.30 Buscar noticias web de un artista.

Este caso de uso forma parte del módulo: **Links.**

Cuando el usuario necesita buscar noticias web de un artista, puede activar esta funcionalidad. La búsqueda puede realizarse sobre artistas que hay dados del alta en sistema o cualquier otro.

5.30.1 Ficha de caso de uso

CASO DE USO	***Buscar noticias web de un artista***		
Versión	1	**Fecha**	30/08/06
Descripción	Buscar noticias web con información de un artista.		
Actores	Usuario		
Precondición	Usuario con sesión activa		
Flujo principal	1 Introducir o escoger nombre artista. 2 Buscar noticias web de artista en http://news.google.es 3 Mostrar resultados		
Flujos alternativos	Si falla la conexión con google se avisa al usuario.		
Postcondición	Muestra los diez primeros resultados de la búsqueda en http://news.google.es		
Comentarios	Utilizar técnicas RSS.		

Tal como se comenta en el comentario del caso de uso la técnica utilizada para obtener la información del artista es RSS (más información,Técnicas utilizadas, RSS).

5.30.2 Prototipo interface

Ver interficie del apartado anterior.

5.30.3 Diagrama de colaboración simplificado

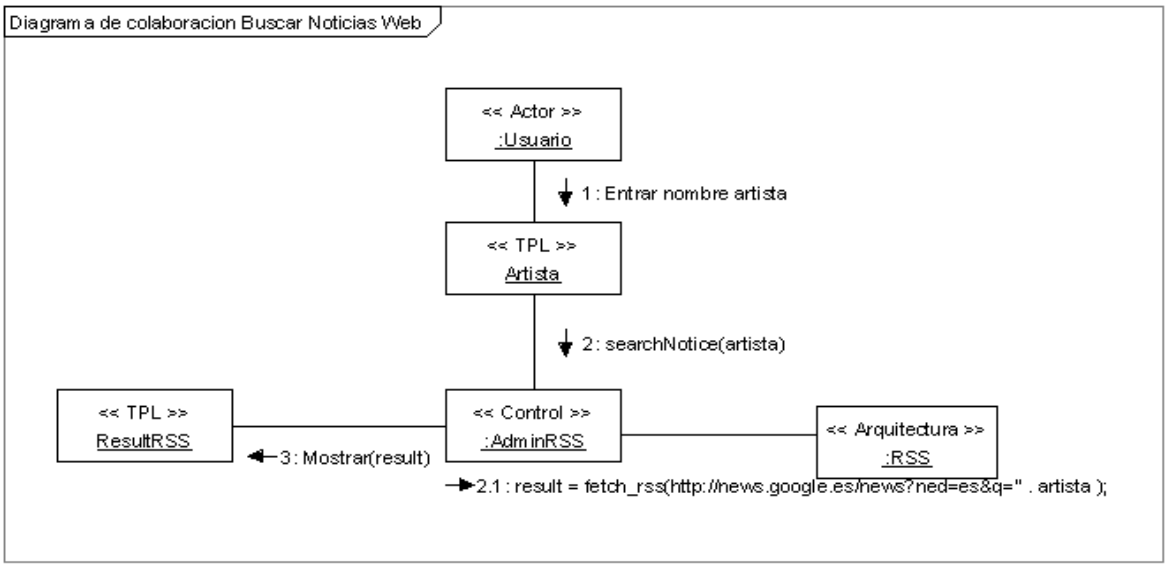
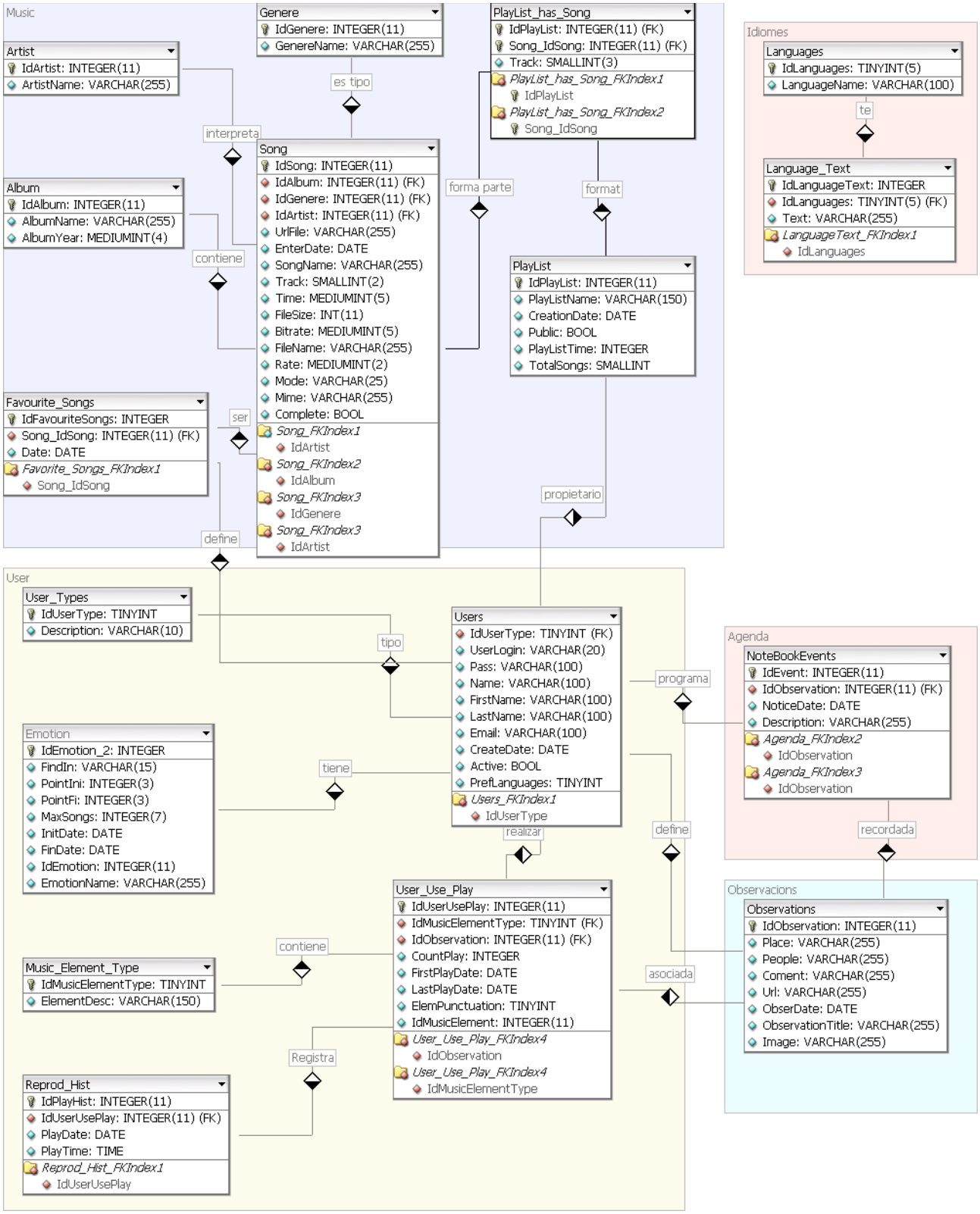


FIG 50.Diagrama de colaboración

Una vez se realiza la llamada al controlador *AdminRSS (mensaje 2),* este llama una funcionalidad de arquitectura (*mensaje 2.1*) que devuelve las paginas donde hay noticias web. Por último hay que mostrar este resultado por pantalla (*mensaje 3*)

5.31 Modelo entidad-relación

Analizando los casos de uso el modelo entidad relación resultante:



6 Estudio Tecnologías Y Herramientas

Una vez se ha presentado la metodología de trabajo y se ha realizado el análisis y diseño, se realizará una explicación de las tecnologías utilizadas para el desarrollo de las características de la aplicación. Antes de nada se va a definir el lenguaje de programación con el que se va a desarrollar la aplicación . A partir de lenguaje se han seleccionado las tecnologías más adecuadas para realizar un desarrollo eficiente , práctico, modular y conseguir un nivel alto de producción.

6.1 Lenguaje de programación

Actualmente existen varios lenguajes de programación web J2EE,.NET, php, entre otros

6.1.1 Posibles alternativas

A continuación se presenta una introducción y algunos de los servicios que ofrecen para el desarrollo de aplicaciones web:

J2EE (Java 2 Enterprise Edition)

J2EE esta formado por un conjunto de especificaciones diseñadas por SUN que permiten la creación de aplicaciones empresariales, por ejemplo: Acceso a base de datos (JDBC) utilización de directorios distribuidos (JNDI), acceso a métodos remotos (RMI/CORBA), funciones para el correo electrónico (JavaMail),uso de Beans y para nuestro caso aplicaciones Web (JSP y Servlets).

Java es un lenguaje orientado a objetos. Sus principales características son: robustez, seguridad, interpretado, simple, distribuido, multitarea y dinámico. Un inconveniente es la utilización de una máquina virtual que interpreta los Java Bytecodes generados por el compilador de Java. Este factor hace ralentizar el sistema. A favor juega que la máquina virtual esta disponible en multiples plataformas y por lo tanto es posible de portar las aplicaciones a diferentes sistemas operativos y servidores.

La *JSP***,** es una tecnología Java que permite a los programadores generar contenido dinámico para web, en forma de documentos HTML, XML, o de otro tipo. Las JSP's permite al código Java y a algunas acciones predefinidas ser incrustadas en el contenido estático del documento web.

La principal ventaja de *JSP* frente a otros lenguajes es que permite integrarse con clases Java (.class) lo que permite separar en niveles las aplicaciones web, almacenando en clases java las partes que consumen más recursos así como las que requieren más seguridad, y dejando la parte encargada de formatear el documento html en el archivo jsp.

Los *JSP* no se puede considerar un script al 100% ya que antes de ejecutarse el servidor web compila el script y genera un servlet, por lo tanto no deja de ser una aplicación compilada.

La tecnología JSP, está teniendo mucho peso en el desarrollo web profesional (sobre todo en Intranets).

Una de las desventajas de Java es que la mayoría de las API's que proporciona Java son muy

genéricas y en la mayoría de los casos es necesario combinar muchas clases para obtener el resultado esperado. Este echo ralentiza el proceso de producción en entidades pequeñas.

Hay varios tipos de herramientas de trabajo que facilitan y aceleren el proceso de producción. Las herramientas disponibles son:

- *IDE*(Integrated Development Environment): Eclipse SDK,Forte, Visual Age, JDeveloper, ...
- *ORM (ver capítulo 6.3)*: Hibernate
- Struts,Maverick: Soporte para el desarrollo de aplicaciones Web bajo el patrón MVC
- Servidores Web: Tomcat

.NET

.NET es un proyecto de Microsoft para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma y que permita un rápido desarrollo de aplicaciones. Basado en esta plataforma, Microsoft intenta desarrollar una estrategia horizontal que integre todos sus productos, desde el Sistema Operativo hasta las herramientas de mercado.

Dentro de la plataforma se encuentran los siguientes entornos: Visual Basic .NET, C#, o Visual J++. Una de la principales características es la interoperabilidad de los diferentes lenguajes soportados, con ello se pueden generar componentes con un lenguaje y introducirlos en otra aplicación escrita en otro lenguaje.

Como Java, *.NET* dispone de un lenguaje especifico para la programación Web, el ASP.NET (Active Server Pages). Es una evolución de las aplicaciones de servidor sobre ASP. La diferencia es que el ASP antiguo es un script interpretado y el *.NET* es un programa compilado. Para el acceso a los datos (principalmente bases de datos) dispone de la tecnología ADO.NET (ActiveX Data Objects). Este acceso principalmente esta dirigido para aplicaciones Web.

Hay varios tipos de herramientas de trabajo que facilitan y aceleren el proceso de producción. Las herramientas disponibles son:

- *IDE:* Visual Studio .NET
- *ORM (ver capítulo 6.3):* NHibernate.
- Servidores Web: Internet Information Services (IIS)

PHP (Hypertext Preprocessor)

PHP es un lenguaje interpretado usado para la creación de aplicaciones para servidores, o creación de contenido dinámico para sitios web. La principal característica son: se ejecuta en el servidor, facilidad de uso, soporta gran cantidad de bases de datos, permite generar páginas con contenidos dinámicos, puede ser ejecutado en diferentes sistemas operativos. Otra ventaja muy importante es la capacidad de expandir su potencial utilizando la enorme cantidad de modulos. disponibles en Internet.

Desde la versión 4.0 php soporta el manejo de objetos con alguna limitación. Actualmente existe

la versión 5.0 que mejora las limitaciones. Esta versión permite herencia, polimorfismo,abstracción de objetos, constructores, destructores, sobrecarga de métodos y otras característica propias de lenguajes diseñados para trabajar con objetos.

Hoy día en la industria Web existe una arquitectura que permite crear aplicaciones económicas, fiables, escalables y seguras, es llamada la arquitectura *LAMP* formada por un servidor Linux,servidor web Apache,base de datos MySQL y PHP.

Hay varios tipos de herramientas de trabajo que facilitan y aceleren el proceso de producción. Las herramientas disponibles son:

- *IDE:* PHPEclipse SDK, Visual PHP.
- Arquitectura *LAMP*
- ORM (ver capítulo 6.3): Propel, Metastorage y PHP Object Generator,...
- MVC Framework using PHP5
- *Smarty*: Es un motor de plantillas para PHP, cuyo objetivo es separar el contenido de la presentación en una página web.
- PEAR (ver anexo 8).

6.1.2 Alternativa seleccionada

Una vez vistas las tecnologías disponibles, la opción escogida a sido PHP, versión 5.0.

PHP se adapta a las característica del proyecto ya que esta soportado por múltiples herramientas que ayudaran a la producción de la aplicación. Además estas herramienta son de uso libre, robustas, y de uso sencillo e eficaz. Dos puntos importantes que han decidido su elección son:

Soporte de objetos con los cuales se creará una aplicación web con componentes reutilizables , escalable y modular. El segundo punto ha sido la popularidad del lenguaje, hay un número considerable de comunidades donde se pueden realizar consultas y encontrar scripts, clases para su reutilización.

J2EE fue descartado porque es un producto especializado para empresas donde hay proyectos de gran envergadura en los cuales participa un equipo con un número elevado de personas. Un segundo motivo fue la complejidad de programación que conllevaría la programación de todos los módulos de aplicación. El nivel de robustez, seguridad y comportamiento dinámico que necesita la aplicación. es cumplido con creces por las característica de PHP, y no son necesarias las característica de Java, además como se ha dicho anteriormente PHP es más flexible y sencillo de comprender que Java.

.NET era una buena opción por ser una tecnología en aumento y por ser también orientada a objetos, pero es una tecnología especifica de Microsoft y hace que limite su uso. Se busca crear una aplicación portable a diferentes plataformas.

6.2 Bases de datos

Una vez se ha escogido el lenguaje de programación se va ha seleccionar la base de datos a utilizar en el proyecto.

En todo proyecto es importante disponer de un almacén de datos potente, robusto, rápido y de fácil acceso. Hoy en día existe una gran variedad de sistemas de gestión de bases de datos (SGBD). Las alternativas que se van estudiar son de licencia libre ya que la naturaleza del proyecto lo requiere (reducir costes). Las posibles opciones comerciales, como pueden ser Oracle y SQL Server, no se ajustan a la característica del proyecto, tanto por el software necesario como por la licencia comercial, estas son adecuadas donde el volumen de datos es mayor y un mayor numero de transacciones.

Actualmente los SGBD de código libre ofrecen el mismo nivel de calidad que los sistemas comerciales más conocidos.

6.2.1 Posibles alternativas

Las opciones que se van a estudiar son: MySQL, Postgressql, SAP DB y Oracle Database 10g XE

MySQL

MySQL se disputa con PostresSQL el mercado de los SGBD de codigo abierto.

Actualmente es una de las bases de datos más utilizadas en el mundo del software libre y sobretodo en las aplicaciones web. Los factores que han ayudado a diferenciar MySQL del resto de SGBD son su rapidez y un coste mínimo de recursos en memoria RAM y CPU.

La rapidez y el poco consumo de CPU conlleva una falta de funcionalidades, algunas de ellas son: no permite procedimientos almacenados (soportados en la versión 5 pero con un comportamiento inestable) y la integridad referencial se deja en manos del programador de la aplicación.

Las funcionalidades más destacadas de MySQL son:

- Soporte para replicaciones (un *master* actualiza múltiples *slaves*)
- Librería para uso inmerso
- Cache de cercas
- Seguridad del sistema, dispone de soporte SSL (Secure Socket Layer)
- Soporta transacciones si se utiliza como motor de almacenamiento InnoDB
- Compatibilidad ANSI SQL 99
- Herramienta de administración web.

Es un SGBD que esta disponible para diferentes plataformas como son: Linux, Microsoft Windows, FreeBSD, Sun Solaris o IBM's AIX.

PostgresSQL

Es uno de los sistemas con mayor experiencia en el mercado (creado 1995) y más conocido en el mundo del software libre.

Desde su nacimiento en 1995 PostresSQL se ha convertido en la base de datos más utilizada en multitud de proyectos, proporcionando al usuario algunas de las prestaciones del mismo nivel que pueden ofrecer los SGBD comerciales, como son: Oracle, Informix, Interbase o SQL Server.

PostrgresSQL esta orientado a intentar dar un servicio a la altura de los mejores SGBD comerciales, ya que esta encarado al uso comercial. A diferencia de MySQL si soporta transacciones, la integridad referencial, indices y vistas. Estas funcionalidades comportan que no sea una base de datos tan rápida (hay algunos estudios que demuestran que es hasta dos o tres veces más lenta) como MySQL, y consume más recursos que MySQL.

Las funcionalidades más destacadas de PostrgresSQL son:

- Soporte de transacciones
- Subconsultas
- Soporte de vistas
- Integridad referencial
- Herencia de tablas
- Tipos definidos por el usuario
- Columnas como vectores que pueden almacenar más de un valor
- Añadir campos en las tablas en tiempo de ejecución
- Funciones de agregación (com sum() o count()) definibles por l'usuario
- Triggers, Comandas SQL que tienen que ser ejecutadas al actuar sobre una tabla.

SAP DB

Es el SGBD de la empresa de software de gestión empresarial SAP. Durante mucho tiempo la empresa dispuso de una versión empresarial, pero en abril del 2001 decidió sacar la versión con licencia GPL. Desde entonces todo el desarrollo de SAP DB ha estado dirigido con licencia libre.

SAP DB es una base de datos muy potente, ya que el hecho de que provenga de un entorno muy especializado, aplicaciones SAP, no esta extendido en el mundo del software libre. Al ser una herramienta tan potente hace que que sea una muy buena opción para proyectos de una cierta envergadura.

Las funcionalidades más destacadas de SAP DB son:

- Soporta outer joins.
- Soporte de rols de usuario
- Vistas actualizadas
- Transacciones y bloqueos implícitos
- Cursores scrollables
- Procedimientos almacenados

Oracle Database 10g XE (Express Edition)

Este SGBD es la versión libre ofrecida por la compañía Oracle. En 2005 presentó la versión beta y en Febrero de 2006 actualizó a la versión estable.

Oracle con este SGBD quiere dar a conocer sus servicios a desarrolladores web,que trabajan con PHP, Java, .NET, XML y aplicaciones de código abierto, estudiantes que empiezan ha aprender conceptos de bases de datos o empresas que quieren ofrecen independencia de software y hardware en sus productos.

Este sistema permite construir aplicaciones web y para poner aprueba las funcionalidades de Oracle, y cuando las dimensiones de la aplicación necesiten un SGBD más potente poder migrar a las versión comercial de Oracle sin ningún coste adicional de tiempo y esfuerzo.

Puede ser instalada en cualquier maquina servidora, pero con los inconveniente de que únicamente puede almacenar hasta 4GB de datos de usuario, utilizar 1GB de memoria RAM, no utilizar mas de una CPU y en su interior no dispone la maquina virtual de java.

Las funcionalidades más destacadas de Oracle Database 10g XE son:

- Utilización de una base de datos profesional
- Escalable y eficiente según las necesidades de cada aplicación.
- Procedimientos almacenados
- Funciones
- Triggers
- Opciones avanzadas de seguridad
- Herramienta para la monotorización de las actividades de la base de datos,
- Administración de usuarios, almacenamiento y memoria.

El sistema operativo Ubuntu (licencia GPL), lleva incorporado en su paquete de instalación este SGBD.

6.2.2 Alternativa seleccionada

Después de la presentación de los posibles SGBD a utilizar el elegido ha sido MySQL, versión 5. Su elección ha sido sobretodo por las necesidades del proyecto (aplicación de uso particular con arquitectura sencilla). Es necesario un SGBD sencillo de instalar con un consumo reducido de memoria RAM y unos recursos de CPU mínimos. Además MySQL forma parte de la arquitectura *LAMP*. con lo cual dispondremos de un sistema económico, fiables, escalables y seguro.

Los otros tres SGBD presentados han sido descartados porque están encarados a un uso comercial donde el volumen de datos es más significativo. Además son sistemas que consumen CPU y memoria RAM, factores negativos para las características del proyecto. Además las funcionalidades extras que aportan de más, en comparación con MySQL, no seria utilizadas, ya que el proyecto no requiere procedimientos almacenado, vistas y la gestión de la integridad referencial no es compleja. En el capítulo 6.4 se da ha conocer una herramienta que soluciona la gestión de la integridad referencial.

6.3 Servidor Web

En este apartado se realiza una explicación del servidor web utilizado y sus principales características.

El servidor utilizado para ejecutar el código PHP ha sido Apache, versión 2. No ha sido necesario estudiar ningún otro servidor web ya que actualmente Apache es la mejor opción que existe en el mercado.

A continuación se realiza una introducción al servidor Apache:

El servidor Apache se desarrolla dentro del proyecto HTTP Server de la Apache Software Foundation.

Apache presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido. Puede ser instalado tanto en maquinas Linux como Windows. Un inconveniente es que no dispone de una interfaz gráfica que ayude en su configuración

Como se ha explicado en el capítulo de lenguajes de programación (6.1) Apache forma parte de la arquitectura LAMP ideal para desarrollar aplicaciones web.

Apache es el servidor HTTP más utilizado en Internet, siendo el servidor HTTP del 70% de los sitios web en todo el mundo.

6.4 ORM (Object-relational mapping)

Uno de los objetivos principales del proyecto es diseñar un modelo de datos que representan objetos de la vida real.

El SGBD MySQL no soporta el almacenamiento de objetos, por lo tanto es necesario un herramienta que permita el mapeo de atributos entre una base de datos relacional y el modelo de objetos de una aplicación. Esta es la funcionalidad de los ORM. Esta técnica es considerada una de las mejores en el diseño de aplicaciones utilizando patrones de diseño, concretamente se trata del patrón *Data Access Objects (DAO)(ver capítulo 7.7.3.2.).*

6.4.1 Posibles alternativas

Es posible construir tu propio ORM, según sean las necesidades del proyecto, aunque este camino no es factible ya que hoy día el mercado ofrece una variedad de ORM's que aportan muy buena solución a la persistencia de objetos. Para cada lenguaje de programación existe una variante de ORM, por ejemplo para Java,Hibernate y para .NET, NHibernate.

Para PHP, lenguaje seleccionado para desarrollar el proyecto, el abanico de posibilidades es enorme ya que PHP es el lenguaje de programación más popular en la comunidad de desarrolladores de software libre. Algunos de los sistemas ORM son: The Propel Project, Metastorage y PHP Object Generator.

The Propel Project

Propel proporciona una gran cantidad de servicios para realizar la persistencia de objetos . Incluye una herramienta para realizar todo tipo peticiones a la base de datos, funciones *set* y *get* mediante una API que permite almacenar y realizar peticiones de datos de forma intuitiva y lógica.

Como buen ORM Propel se encarga de crear las sentencias SQL y realizar las conexiones con la base de datos, de manera que solo hay que preocuparse de conocer las clases y objetos para acceder a los datos de manera cómoda y sencilla. Véase un ejemplo:

```
$book = BookPeer::retrieveByPK(5);
$book->setTitle('The Propel Story');
$book->save();
```

Para construir los objetos simplemente hay que definir un fichero XML y Propel se encarga de generarlos incluyendo un fichero .sql con la definición de la base de datos.

Una de las principales característica es la emulación de llaves foráneas aunque el SGBD no lo soporte,MySQL es un ejemplo. Esto no significa que perdamos el controlar de la base de datos, todo lo contrario Propel proporciona suficientes características que permiten controlar todas las operaciones realizadas en la base de datos.

Las características más destacadas de Propel son:

- Reducir el tiempo de desarollo de la aplicación

- Fácil ampliación del modelo de datos
- Posibilidad de cambiar el comportamiento de las clases según necesidades
- Herramientas intuitivas para construir sentencias SQL y más.
- Independencia de base de datos
- Intuitivo, predecible, fácil de utilizar
- Soporta herencia de clases
- Emulación llaves foráneas
- Completo conjunto de operaciones para manipular los atributos de una base de datos

Las funcionalidades explicadas anteriores son posibles a una mezcla de tecnologías propias de Propel: *Generator* y *Runtime Framework* , y otras tecnologías: *Phing*, *Creole* y *PEAR::Log (ver Anexo 6)*

Metastorage

Metastorage es una aplicación que genera código automáticamente para crear una API Orientada a Objetos que permite guardar, obtener y manipular la persistencia de objetos. Estos objetos forman parte del diagrama de clases de alto nivel. Las clases generadas serán escritas en PHP

Metastorage permite un proceso de desarrollo eficaz para aplicaciones de tamaño mediano o grande que necesitan almacenar y obtener datos de una base de datos utilizando sentencias SQL. No solo base de datos también permite otro tipo de contenedores de persistencia como son fichero XML o servidores LDAP.

Los objeto son generados a partir de la definición de un fichero XML basado en el formato CPML (Component Persistence Markup Language, más información, www.meta-language.net/metastorage.html) .

Para poder acceder, manipular y guardar los objeto hay que realizar 4 pasos:

1. Establecer la conexión con la base de datos
2. Iniciar la transacción (Acceso a datos)
3. Finalizar la transacción
4. Cerrar la conexión con la base de datos.

Ejemplo de uso:

```
$factory->initialize() (1)
      $factory->begin() (2)
            $writer = &$factory->getwriter(5);
            $writer->name = 'Sthepen king';
            $success= $writer->persistarticle();
       $this->end($success);(3)
```

```
$factory->finalize();(4)
```

Las características más destacadas de Metastorage son:

- Reducir el tiempo de desarrollo de la aplicación
- Producir software de calidad
- Flexibilidad y independencia en los sistemas de persistencia de datos
- La API generada es la misma para diferentes sistemas de persistencia.
- Posibilidad de cambiar el comportamiento de las clases según necesidades
- Las clases generadas permiten instalar o actualizar la base de datos.
- Validación y creación de formularios.
- Generación de diagramas de clase con notación UML.

Más información en http://www.meta-language.net/

PHP Object Generator (POG)

POG ahorra tiempo a los programadores de PHP, se encarga de generar objetos php eficientes y pruebas de test. El tipo de objetos generados permite realizar la persistencia de objetos utilizando el patrón DAO.

Para generar las clases hay que entrar en la pagina web (http://www.phpobjectgenerator.com/) e introducir manualmente la información del modelo entidad-relación.

El código generado es extremadamente limpio, fácil de utilizar y entendedor.

Los SGBD soportados son: MySQL, FireBird, SQLServer, SQLLitey PostresSQL.

Las características más destacadas de PHP Object Generator son:

- Reducir el tiempo de desarollo de la aplicación
- Generación de código limpio con clases de test
- Independencia SGBD
- Operaciones básicas (Create,Retrieve,Update y Delete)
- Genera fichero setup (.sql)
- Genera relaciones padre-hijo (herencia)

6.4.2 Alternativa seleccionada

Una vez presentadas las posibles soluciones se ha optado por utilizar Propel.

Se ha decidido usar este ORM porque los servicios ofrecidos por las clases que genera son más completos que las demás alternativas. Entre ellos se destacan la posibilidad de obtener datos por llave primaria o realización de consultas al SGBD utilizando una clase diseñada para ello sin tener que crear sentencias SQL. La simulación de llaves foráneas ha sido uno servicio que ha tenido mucho valor en la decisión. Con él se pueden crear objeto padre e hijo sin tener ningún tipo de problema. Otro punto a favor ha sido qué Propel esta basado en tecnología estándar de la fundación Apache, lo cual da garantías de calidad, robustez y seguridad a la herramienta..

Metastorage es una herramienta muy potente la con servicios de calidad, casi los mismos que Propel. Pero Metastorage no soporta simulación de llaves foráneas, además la utilización de las clases es más compleja y el proceso de generación de clases conlleva algunas complicaciones. Por otra banda Metastorage no esta basado en ninguna tecnología estándar como es el caso de Propel.

PHP Object Generator no esta al mismo nivel de servicios que Metastorage y Propel, además hay un punto muy desfavorable, cada vez que se desea generar una clase hay que entrar manualmente la información en la web que ofrece el servicio. Esto implica que si se decide cambiar algún atributo de la tabla hay que volver a introducir toda la información.

6.5 Herramientas de trabajo

A continuación se van a explicar las diferentes herramientas utilizadas para el desarrollo, instalación de tecnologías y administración de base de datos.

6.5.1 Desarrollo

La plataforma escogida para trabajar ha sido Windows, concretamente ***Microsoft Windows Xp Professional, Service Pack 2.***

La instalación de *PHP, MySQL y Apache* requiere un grado de complejidad y un tiempo de adaptación considerable, sobretodo si es la primera vez que se tiene contacto directo con estas tecnologías. El tiempo de instalación requerido fue de **36** horas.

Uno de los requisitos del proyecto es que pueda ser instalado y utilizado por usuarios no informáticos. Por ello se buscó una herramienta que permitiera instalar de manera cómoda y sencilla *PHP, MySQL y Apache,* e reducir el tiempo de instalación. La solución encontrada fue *XAMPP*.

6.5.2 Instalación de tecnologías

XAMPP es un software de uso libre que contiene Apache, MySQL y algunas herramientas necesarias para utilizar PHP y Perl. *XAMPP* se encuentra registrado con licencia GPL con lo cual se encuentra libre de costes. Se encuentra disponible para Windows, Linux, Sun Solaris y Mac OS X. En el manual de instalación de la aplicación se detalla el procedimiento de instalación.

XAMPP incluye también otros módulos como son OpenSSL y phpMyAdmin. De estos dos módulos se ha utilizado phpMyAdmin para realizar la administración de la base de datos.

6.5.3 Administración base de datos

phpMyAdmin es una herramienta escrita en PHP que permite administrar MySQL a través de Internet. Permite crear y borrar bases de datos, crear/borrar/alterar tablas, borrar/editar/añadir registros, ejecutar cualquier sentencia SQL y administrar campos clave.

6.5.4 Herramientas de implantación

Para la implantación del proyecto las herramienta utilizadas han sido las siguientes:

- **Poseidon Profesional Edition (versión 4.0.1-0):** Notación UML, diagramas de secuencia, colaboración, diagrama de clases, ...
- **DBDesigner 4 (versión 4.0.5.6 Beta):** Diseño del modelo entidad-relación.
- **Eclipse SDK (versión 3.1.2) +PHPeclipse (versión 1.1.8):** Codificación de las clases PHP.
- **Dreamweaver MX:** Codificación del código HTML y código Smarty para generar

las templates de la aplicación.

- **OpenOffice.org 2.0 Writer:** Herramienta utilizada para redactar la documentación.
- **OpenOffice.org 2.0 Calc:** Herramienta utilizada para calcular las horas de trabajo.
- **Axialis IconWorkshop 5.0 (versión 5.0.2):** Diseño de iconos.

6.6 Aplicaciones estudiadas

Después de conocer las tecnologías y herramientas utilizadas para el desarrollo y implementación del proyecto, se van ha dar a conocer las aplicaciones web que realizan funcionalidades semejantes a las del proyecto. Estas fueron estudiadas para descubrir las carencias de las aplicaciones web de catalogación. Este estudio además de conocer las carencias me permitió familiarizarme con el lenguaje PHP , MySQL y Apache y las técnicas de streaming.

El estudio y las conclusiones del mismo están disponibles en el anexo 9.

Gracias ha este estudio se pudieron definir las característica generales de la aplicación.

El tiempo empleado para el estudió fue un total **40** horas de las cuales **29** fueron dedicadas a instalación y solucionar problemas internos.

A continuación se nombran las tres aplicaciones estudiadas. En el anexo 8 se encuentran los informes realizados.

- **kplaylist (versión 1.6)**
- **ampache (versión 3.3.2-Beta 1)**
- **Jinzora (versión 2.3.7)**

Se estudio una cuarta aplicación que únicamente funciona sobre maquinas GNU-Linux, se probó la versión Demo en la web. De esta aplicación se estudió el código PHP que realizaba el Streaming, el cual fue de utilidad para entender el funcionamiento de un servidor de Streaming.

- **mp3act** (www.mp3act.net)

6.7 SOAP / Servicios Web

En este apartado se explica la tecnología utilizada para buscar páginas web con contenidos de artistas. Para ello se ha utilizado un servicio web de la compañía Google. Para entender como funciona este servicio web primeramente se va hacer una introducción al protocolo SOAP.

6.7.1 Introducción a SOAP

SOAP (acrónimo de Simple Object Access Protocol) es un protocolo estándar creado por Microsoft, IBM y otros, está actualmente bajo el auspicio de la W3C que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. SOAP es uno de los protocolos utilizados en los *servicios Web*.

SOAP es un marco extensible y descentralizado que permite trabajar sobre múltiples pilas de protocolos de redes informáticas. Los procedimientos de llamadas remotas pueden ser modelados en la forma de varios mensajes SOAP interactuando entre sí.

SOAP funciona sobre cualquier protocolo de Internet, generalmente HTTP, que es el único homologado por el W3C y el que ofrece menos problemas ante un firewall. SOAP tiene como base XML, con el diseño estándar,Cabecera-Desarrollo (Header-Body). El Header es opcional y contiene metadatos sobre enrutamiento, seguridad o transacciones. El Body contiene la información principal definida con un *schema XML* propio.

En el anexo 9 es posible observar un ejemplo SOAP.

6.7.2 Servicios Web

Una vez ya conocemos el protocolo SOAP se va a realizar una introducción a los Servicios Web.

Un Servicio Web es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes y ejecutadas sobre cualquier plataforma pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos.

Hay cuatro areas de protocolos que permiten definir, localizar, implementar y permitir a los servicios web interactuar con otros servicios:

- *Servicio transporte:* Encargado de transportar los mensajes entre aplicaciones web.

  Protocolos: *HTTP,SMTP,FTP,BEEP*(Blocks Extensible Exchange Protocol)

- *Mensajes XML:* Responsable de codificar los mensajes en formato XML para que puedan ser entendidos por cualquier conexión.

  Protocolos: *XML-RPC, SOAP y REST* (Representational State Transfer) .

- *Descripción del servició:*Describe la interfaz pública de un servicio web.

  Protocolo: *WSDL* (*Web Services Description Language)*

- *Descubridor de servicios:* Es un servicio centralizado que permite publicar la localización y descripción de los servicios web.

  Protocolo: *UDDI* (*Universal Description, Discovery, and Integration*)

Algunas ventajas de los servicios web:

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Los servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Al apoyarse en HTTP, los servicios Web pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado.

Algunos aspectos negativos:

- Su rendimiento es bajo si se compara con otros modelos de computación distribuida, tales como RMI, CORBA, o DCOM.
- Existe poca información de servicios web para algunos lenguajes de programación
- Al apoyarse en HTTP, pueden esquivar medidas de seguridad basadas en firewall cuyas reglas tratan de bloquear o auditar la comunicación entre programas a ambos lados de la barrera.

6.7.3 Google SOAP Search API (Beta)

Una vez se han comprendido los conocimientos de *SOAP* y Servicios Web se presenta el Servicio Web que permite buscar páginas web con contenidos de un artista, *Google SOAP Search API.*

El servicio *Google SOAP Search API* permite desarrollar aplicaciones que realicen consultas al buscador Google y mostrar el resultado de la búsqueda en tus propias aplicaciones. Google utiliza los estándares *SOAP* y *WSDL* así el Servicio Web queda independiente del lenguaje de programación que este utilizando el programador de la aplicación.

Las peticiones posibles son:

- *doGetCachedPage*: Petición de páginas en la cache de google.
- *doGoogleSearch*: Petición de búsqueda de contenidos.
- *doSpellingSuggestion*: Petición de sugerencias en la ortografía de la búsqueda de contenidos.

La petición utilizada en el proyecto ha sido *doGoogleSearch.* En el *anexo 9* se muestra un ejemplo de la estructura del fichero XML para realizar la petición, y el resultado obtenido de la petición.

6.7.4 Implementación del Servicio Web

A continuación se hace una explicación de como se ha utilizado el servicio en el proyecto.

Tal como se ha visto en el apartado de SOAP, la petición y la respuesta son un fichero XML. Para utilizar este servicio no es necesario conocer como hay que construir y leer los ficheros XML. Los lenguajes de programación proporcionan herramientas para ello. En nuestro caso PHP dispone de una clase llamada *SoapClient* que realiza el trabajo de crear, enviar y recibir el fichero XML.

Para llevar acabo la petición SOAP hay que configurar previamente el objeto. Es necesario declarar la URL del Servicio Web y un array con los parámetros requeridos por la petición, en nuestro caso *doGoogleSearch.* Google informa de los parámetros necesarios en:. Una vez se tiene la configuración se realiza la petición y se obtiene un array con el resultado de la operación. El array resultante contiene multitud de información para cada enlace encontrado. La información utilizada para presentar al usuario ha sido:

- Título enlace (*title*)
- Dirección URL (*URL*)
- Descripción (snippet)

Modelo de ejemplo Anexo 9.

6.8 RSS

A continuación se va a realizar una introducción a la tecnología utilizada para buscar las noticias más actuales de los artistas. Una vez presentada la tecnología se explicará como se ha implementado en el proyecto.

6.8.1 Introducción RSS/Atom

RSS/Atom son parte de la familia de los formatos XML (ver ejemplo anexo 11) desarrollados específicamente para todo tipo de sitios que se actualicen con frecuencia y por medio de los cuales se puede compartir la información y usarla en otros sitios web o programas (sindicación). Estos formato de datos son llamados ***feed.***

Un *feed* es el resumen, actualizado regularmente, de un determinado contenido web y los vínculos a la versión completa del mismo. Al suscribirse al *feed* de un sitio web mediante un lector de feeds, obtendrá un resumen del contenido nuevo de dicho sitio.

El acrónimo *RSS* se usa para los siguientes estándares:

- **Rich Site Summary** (RSS 0.91)
- **RDF Site Summary** (RSS 0.9 y 1.0)
- **Really Simple Syndication** (RSS 2.0)

6.8.2 Feeds de Noticias

Una vez se conoce que es un *feed* se presenta la fuente de *feeds* utilizado en el proyecto, este ha sido Google.

Google tiene acceso a unos 4.500 fuentes actualizadas continuamente para el dominio news.google.com y 700 en news.google.es. El trabajo de Google llega más allá, Google es un generador de feeds. Este permite introducir un tema concreto para una noticia,ejemplo madonna, el resultado es una colección de feeds relacionados con el tema Los formatos XML soportados por google son RSS 2.0 o Atom 0.3, según él crea oportuno. Para poder acceder a estos feeds es necesario de disponer de un lector de feeds, e aquí donde entra el proyecto. En el siguiente apartado se explica la herramienta utilizada para acceder a los feeds.

6.8.3 Lector de feeds

Para tener acceso a los feeds de google ha sido necesario implementar un lector de feeds. Para ello se ha utilizado una classes especial de lenguaje PHP.La herramienta utilizada es ***Magpie RSS.***

Magpie RSS es simplemente una interface de PHP que permite parsear feeds RSS. Una vez se parsea el RSS se obtiene un array resultante con la información del feed. La información que se ha utilizado para presentar al usuario final ha sido: *description.* Description tiene el título de la notícia,descripción, URL original y en algunos casos contiene una fotografía. El anexo 11 muestra un

ejemplo de uso.

6.9 Framework MVC

En este apartado se hace una breve introducción a la herramienta más importante utilizada, el framework MVC (FMVC). Es considerada la más importante ja que el diseño , desarrollo y utilización de los objetos (clases) siguen la metodología de trabajo propuesta por el FMVC. Siguiendo esta metodología se ha obtenido una aplicación altamente escalable y modular.

El FMVC es una herramienta de código abierto desarrollada en lenguaje PHP. Está diseñada para implementar el patrón Model View Controller (MVC) en proyectos que utilizan el lenguaje PHP. Estos proyectos tienen que utilizar la versión 5.0 de PHP ya que el FMVC esta diseñado para trabajar íntegramente con programación orientada a objetos (POO) e la versión 5.0 es la única que los soporta.

El FMVC no solo permite la implementación del MVC sino que soluciona dos problemas muy comunes en las aplicaciones web: los permisos de acceso de usuario para cada página de la aplicación y gestión de conexión con la base de datos que es necesaria en la mayoría de páginas.

Los componentes que ofrecen las funcionalidades descritas anteriormente son los siguientes: un grupo de *clases base*, un *fichero index.php*, un *fichero config.php*, un *directorio de módulos, módulos sencillo y views* (vistas).

Las ***clases base*** controlan el acceso de usuarios y gestionan la conexión con la base de datos. Las clases bases están agrupadas en un directorio llamado *includes*. Algunas de las clases base son: FR_Object, FR_Auth, FR_Session, FR_Object_DB.

Fichero index.php: Componente principal del FMVC. Sus funcionalidades son :

1) Procesar la petición que llega a través de la URL.

2) Crear y llamar al objeto que responde a la petición.

3) Escoger el tipo de view para el usuario final.

Fichero config.php: Fichero donde se definen las variables globales de la aplicación.

Directorio de módulos: Directorio dónde se agrupan los distintos módulos sencillos que forman parte de la aplicación.

Módulos sencillos: Cada uno de los módulos de la aplicación que implementa una parte del modelo de negocio. Cada modulo es responsable de conocer las clases necesarias para dar servicio a su funcionamiento. No solo las clases sino las plantillas Smarty que son necesarias para presentar al usuario los resultado de las operaciones realizadas en las clases.

Views: Son los posibles formatos que ofrece el framework para presentar al usuario los resultados de las operaciones realizadas en un modulo sencillo. Los formatos son: *ficheros XML*, *plantillas Smarty* (más información 6.10) y *debug*. Estos formatos no eran suficientes para cumplir los requisitos del proyecto, por ello se incluyo un nuevo formato, *ficheros playlist*. Este formato se encarga de construir un fichero playlist.

Más detalles sobre el framework en el capítulo 7.4.

6.10 Smarty

Smarty es un tipo de ingeniería de template escrita en lenguaje PHP. Smarty separa el código PHP , como por ejemplo el del proceso de negocio, del HTML y genera contenido web utilizando sus propios tags.

Utilizando Smarty se ha conseguido tener una aplicación con un código limpio y mucho más flexible a modificaciones.

6.11 getID3()

getID3 es una herramienta de código abierto, disponible para todo tipo de plataformas (cross-platform), concretamente es una librería para ser utilizada con el lenguaje PHP.

El tipo de información obtenida incluye tiempo de reproducción, metadatos, bitrate, sample rate, número de canales (stereo/mono) y mucho más. No sólo lee sino que permite escribir metadatos (tags,más información anexo 1) de diferentes formatos.

Más información; www.getid3.org.

6.11.1 Implementación en el proyecto

Antes de entrar en detalles de como se ha utilizado esta herramienta se explicará el funcionamiento de getID3. De esta manera se entenderá mejor el tipo de implementación realizada.

Funcionamiento getID3

Para poder analizar el archivo es necesario informar a getID3 de su dirección en el árbol de directorios de la máquina local, una vez informada se efectúa el análisis del archivo. Al finalizar el análisis el resultado es almacenado en un array de resultados. Cada posición del array contiene un tipo de información diferente (metadatos, bitrate, sample rate ...). Según el tipo de archivo que se ha analizado, ya sea de audio, vídeo, imagen ..., el resultado de la información y el acceso ha esta es diferente. El array de resultados necesario para el proyecto corresponde al de los ficheros de audio MP3. Estos ficheros almacenan la información la información en los tags ID3v1 o ID3v2 (más información anexo 1).

Implementación

Como se ha explicado en el parágrafo anterior, getID3 devuelve un array de resultados diferente cuyo acceso a los datos se realiza de manera distinta dependiendo del tipo de archivo analizado. Por este motivo y además pensando en una posible ampliación de la aplicación, la cual incorpore nuevos tipos de archivo de reproducción, se han creado unas clases especiales para minimizar el acoplamiento (patrón *low coupling**) de la herramienta getID3. Sobretodo se ha intentado minimizar el problema con el tipo de formato de audio. El diseño inicial solo utiliza el formato MP3, requisitos del proyecto, pero es posible añadir cualquier otro tipo de formato sin que haya costes adicionales (patrón *factory* + *polimorfisme*). Además de desacoplar el formato de ficheros, se ha desacoplado getID3 de tal modo que si en un futuro existiera una mejor herramienta esta podría ser cambiada sin que el el código principal de la aplicación haya de ser modificado (patrón *facade**).

*En el capítulo patrones de diseño se informa de los patrones de diseño utilizados.

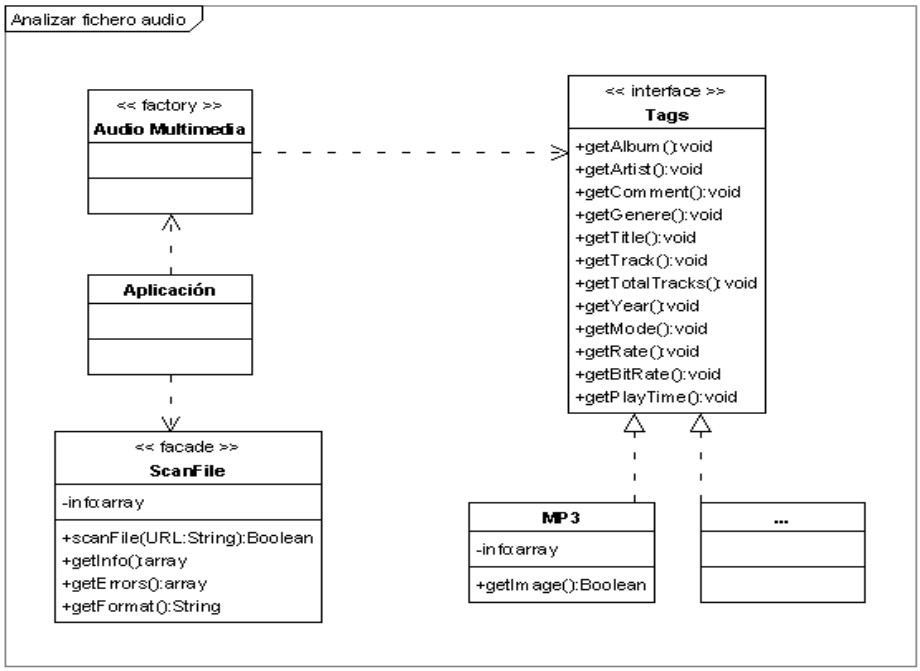


FIG 51: Diagrama de clases

7 Implementación

Una vez realizado el análisis de requerimientos, diseño de la aplicación y presentado las herramientas para el desarrollo, se dispone de todos los elementos necesarios para llevar a cabo la implementación.

La implementación se dividirá en dos sistemas. El *sistema principal* encargado de gestionar la mayoría de funcionalidades descritas en el proceso de análisis y el *sistema secundario* que se encargará de gestionar el servidor de streaming (ver anexo 4).

7.1 Sistema Principal

La implementación del sistema principal seguirá la estructura propuesta por el framework MVC (ver capítulo 6.9). El hecho de utilizar el framework MVC conducirá a implementar los siguientes componentes: Módulos, Includes, Interficies y Base de datos.

7.1.1 Componente Módulos

El componente módulos estará compuesto por catorce *módulos sencillos*.

Cada ***módulo sencillo*** se encargará de responder a las peticiones que lo caracterizan. Los componentes que lo formarán: como mínimo una *clase de control* responsable de procesar y responder a las peticiones, una *clase de constantes* y un *fichero de configuración*, en caso de necesitarlo. Además cada módulo sencillo tendrá una interficie que le permitirá presentar los resultados de las peticiones.

Veamos un ejemplo:

El módulo sencillo agenda estará formado por la clase de control *AdminAgenda* que se encargará de procesar las peticiones para la gestión de eventos de la agenda. Los resultados de la gestión de eventos serán presentados por la interficie *admin_agenda.tpl.*

7.1.2 Componente Includes

El componente includes estará compuesto por *clases de soporte a los módulos*. Existirán tres tipos de clases de soporte: *las clases base, las clases del modelo de objetos* y *las clases con servicios complementarios*.

Las ***clases base*** serán las clases que forman el framework MVC (ver capítulo 6.9).

Las ***clases del modelo de objetos*** representarán el modelo de negocios de la aplicación.

Las ***clases con servicios complementarios*** serán grupos de clases responsables de resolver peticiones específicas recibidas por los *módulos sencillos*. Por ejemplo las clases SOAP (ver capítulo 6.7) o las clases getID3 (ver capítulo 6.11).

7.1.3 Componente Interficies

El componente interficies estará compuesto por dos tipos de interficies, las *interficies de estructura y las interficies de los módulos.*

Las ***interficies de estructura*** serán responsables de construir el esqueleto principal de la aplicación. Existirán dos tipos de estructura:

- Plantillas Smarty (ver capítulo 6.10).
- Ficheros PlayList (ver anexo 3).

Las ***interficies de los módulos*** serán responsables de presentar los resultados obtenidos por las clases de control de los *módulos sencillos*. Este tipo de interficie utilizará únicamente las plantillas Smarty.

7.1.4 Base de datos

El componente base de datos estará compuesto por la implementación de la *base de datos* y la *conexión con la base de datos*.

La ***implementación de la base de datos***, se dispondrá de un SGBD MySQL versión 5.0 que podrá ser instalado en cualquier PC de sobremesa con sistema operativo Windows XP , versión Profesional o Home.

La ***conexión con la base de datos*** será gestionada por un sistema de persistencia, Propel (ver capítulo 6.4.1).

7.2 Sistema Secundario

La arquitectura del sistema secundario ha tomado como modelo los sistemas de streaming estudiados (ver anexo 9).

Para crear este sistema serán necesario los siguientes componentes: Bases de datos, Validador de Datos, Gestor de Logs, Servidor Streaming y Estadísticas. Antes de describirlos se exponen las causas por los cuales se va ha construir un segundo sistema.

La principal causa es la incompatibilidad con el framework MVC (estructura base del sistema principal) y el Servidor de streaming. El servidor de streaming no precisa de interficie de usuario ya que no existe interacción directa con el usuario. Además el framework MVC no permite establecer una comunicación continua y directa entre servidor y cliente. Este tipo comunicación es indispensable para la emisión de los datos de audio. Por último al disponer de un sistema secundario liberará al controlador del framework MVC de peticiones con lo cual se conseguirá un mejor rendimiento del framework MVC.

Una vez conocidos los motivos de implementación del sistema secundario se prestan sus componentes:

7.2.1 Base de datos

Este componente será el mismo utilizado en el sistema principal, con el se comunicará y compartirá la información entre los dos sistemas.

7.2.2 Validador de Datos

Este componente se encargará de validar la petición enviada por el cliente (reproductores multimedia). La validación es realizada por un script. Cuando el script validador localiza un error cancela la operación y se activa el *Gestor de Logs*.

7.2.3 Gestor de Logs

Este componente se encargará de registrar los errores localizados por el *Validador de Datos*. Los registros serán almacenados en un fichero de logs. Este fichero se encontrará en un directorio temporal (/tmp) con el nombre "stream_error.log".

7.2.4 Servidor de Streaming

Será el principal componente del sistema secundario.

Se encargará primeramente de realizar la configuración necesaria para el envío de datos del fichero de audio solicitado por el cliente. La configuración consistirá en definir: Unidad de medida (bytes), Tamaño del fichero, Tipo mime y Información musical del fichero solicitado (Título canción, Nombre artista y formato MP3)

Segundo y último el servidor procederá a la lectura y envío de los datos de audio almacenados en el fichero MP3, acción de streaming.

El proceso de lectura consistirá en lecturas de 8192 bytes de datos del fichero MP3 y seguidamente enviar la información al cliente. Este proceso se repetirá hasta enviar todos los datos del fichero solicitado.

7.2.5 Estadísitcas

Este componente será el encargado de registrar en la base de datos, usuario, fichero solicitado , fecha y hora de la petición.

7.3 Arquitectura Física

Una vez conocida la implementación, la arquitectura física de los dos sistema será la siguiente:

En ambos sistemas la *codificación* será realizada en lenguaje PHP, tal y como ya se ha dicho en apartados anteriores.

7.3.1 Arquitectura del Sistema Principal

Los componentes del sistema principal quedarán organizados de la siguiente forma:

Raíz (/):

Directorio principal.

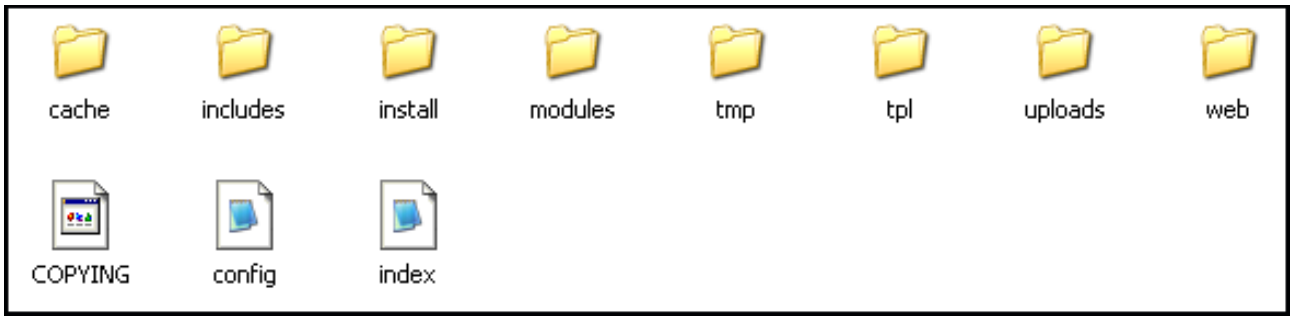


FIG 52: Estructura del directorio

- *Cache:* Directorio utilizado por la ingeniería de plantillas Smarty.
- *Includes:* Conjunto de archivos que formarán el C*omponente Includes* (ver capítulo 7.1.2).
- *Install: A*rchivos necesarios para la instalación.
- *Modules:* Conjunto de archivos que forman el *Componente Modules* (ver capítulo 7.1.1).
- *Tmp:* Guardará el fichero de logs.
- *Tpl:* Contiene los archivos para con las *Interficies de Estructura*.
- *Uploads:* Guarda las fotografías subidas al sistema por los usuarios (ver capítulo 5.11).
- *Web*: Archivos necesarios para construir las interficies de estructura. Hojas de estilo (Css), scripts de JavaScript y imágenes.
- COPING: Es la licencia del framework MVC.
- config: Fichero con las configuraciones del sistema.
- index: Fichero con el código fuente del controlador del framework MVC (ver capítulo 6.9).

/Modules:

El *Componente Modules* quedará organizado de la siguientes forma:

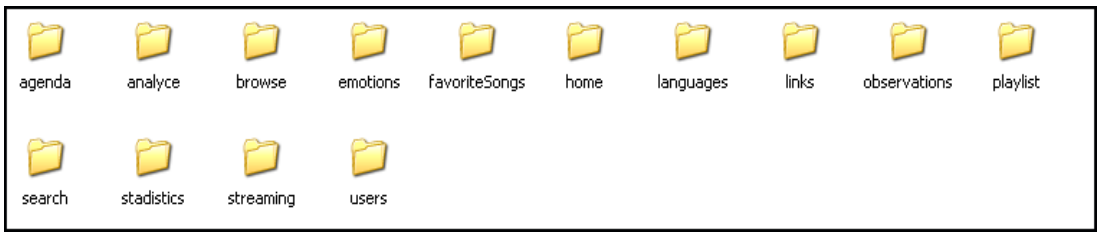


FIG 53: Estructura del directorio

Cada uno de los directorios representará un *módulo sencillo*. Los módulos sencillos han sido obtenidos en el proceso de análisis (ver capítulo 5).

/modules/agenda/ :

Ejemplo de *módulo sencillo*:

FIG 54: Estructura de directorios

- *tpl*: Directorio con las interficies del módulo sencillo.
- *AdminAgenda*: Archivo con el código fuente de la clase controladora del módulo sencillo Agenda. El nombre del archivo y la clase controladora siguen el patrón: "Admin" + Nombre módulo sencillo. Todos los nombres de archivo y clase seguirán el mismo patrón.
- *ConstAgenda*: Archivo con la clase que definirá las constantes para el controlador. El nombre del archivo y la clase controladora siguen el patrón: "Const" + Nombre módulo sencillo. Todos los nombres de archivo y clase seguirán el mismo patrón.
- *config*: Archivo de configuración del propio módulo sencillo.

/includes:

El *componente includes* quedará organizado de la siguientes forma:

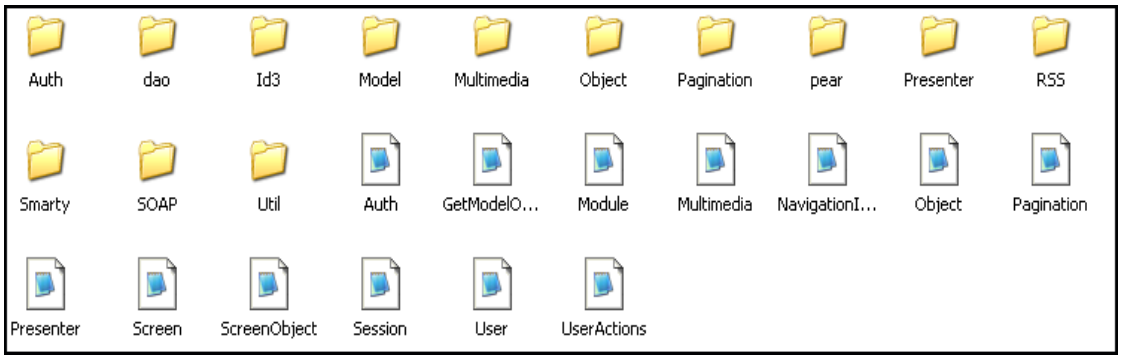


FIG 55: Estructura de directorios

Los directorios de la imagen organizarán los archivos con el código fuente necesario para ejecutar una función especifica de soporte a los *módulos sencillos*.

Los archivos de la imagen, al igual que los archivos de los directorios, contienen el código fuente necesario para ejecutar una función especifica de soporte a los *módulos sencillos*.

- ***Clases del modelo de objetos:***

  Organizadas en el directorio: *Model.*

- ***Clases base:***

  Organizadas en los directorios: *Auth, Object, Presenter*
  Archivos*: Auth, Module, Object, Presenter, Session, User*

- ***Clases con servicios complementarios:***

  Código fuente de los directorios: Id3, Multimedia, Pagination, pear, RSS, SOAP, Util.
  Archivos: GetModelObject, Multimedia, NavigationInfoUser, Pagination,Screen. ScreenObject, UserActions.

/includes/dao/:

El directorio dao organizará el código fuente que forma parte del sistema de persistencia Propel.

7.3.2 Arquitectura del Sistema Secundario

Los componentes del sistema secundario quedarán organizados de la siguiente forma:

El componente *Servidor Streaming* se organizará dentro del *módulo sencillo* streaming, ***/modules/streaming/***. El nombre del fichero que identificará el componente será ***play***.

El componente *Gestor de logs será* miembro de las *clases con servicios complementarios,* ***/includes/util/***. El nombre del fichero que identificará el componente será ***Logs*** .

El componente *Estadística* al igual que el Gestor de Logs será miembro de las *clases con servicios complementarios,* ***/includes/***. El nombre del fichero que identificará el componente será ***UserActions***.

El componente *Base de datos* será el mismo utilizado en el sistema principal, tal y como ya se ha dicho anteriormente.

7.4 Interacción de componentes

Llegados ha este punto, conocedores de los componentes involucrados en los dos sistemas, se realizará una explicación para comprender la interacción que existe entre ellos.

7.4.1 Interacción Sistema principal

Los componentes del sistema principal serán utilizados durante dos fases: "*Fase de Escucha*" *y* "*Fase respuesta de peticiones*"*.*

Fase de Escucha (Request)

Todas las peticiones del sistema principal pasarán por un punto de entrada, *el controlador.* El controlador será el componente más importante de las *clases base.* El controlador será el responsable de seleccionar el *módulo sencillo* que resolverá la petición. Una vez haya localizado el *módulo sencillo* creará el objeto que resolverá la petición. Para seleccionar el *módulo sencillo* apropiado y crear el objeto correcto el controlador recibirá las siguientes variables de entrada:

- *module:* Informa del módulo sencillo que resuelve la petición.
- class: Informa del nombre de la clase controladora que resuelve la petición.
- *event*: Informa del nombre de la función miembro de la clase controladora que resuelve la petición.

Es imprescindible que las variables lleguen al controlador a través de la URL o a través del método GET de los formularios HTML.

En caso de necesitarse variables auxiliares para resolver la petición podrán ser añadidas sin ninguna restricción.

Fase de Respuesta (Response)

El objeto de respuesta tomará el control y llamará a la función miembro que ha sido declarada con el valor recibido por la variable de entrada *event.*

Para solucionar con éxito la petición el objeto utilizará los componentes del C*omponente Includes* que le sean necesarios.

Una vez el objeto a solucionado la petición el controlador tomará de nuevo el control. Esta vez se encargará de crear el objeto que presentará los datos obtenidos en el proceso de resolución. El objeto será del tipo I*nterficie de Estructura.*

Para crear el objeto de interficie adecuado el controlador consultará la propiedad "*presenter*"*.* Esta propiedad se encontrará definida en el objeto que ha solucionado la petición.

7.4.2 Interacción Sistema secundario

Los componentes que lo formarán interactuarán de la siguiente forma:

Primeramente se establecerá la comunicación con la base de datos, será utilizado el mismo componente de conexión que es utilizado en el sistema principal.

Segundo entrará en funcionamiento el *Validador de Datos*, se encargará de validar los datos de entrada recibidos en la petición. Estos son: Identificador de canción, usuario que realiza la petición y información musical de la canción. En caso de detectarse un error en los datos de entrada se cancelará la petición y se disparará el *Gestor de Logs.*

Tercero el servidor de streaming se encargará de emitir la canción solicitada.

Por último cuando el servidor de streaming ha finalizado de emitir todos los datos se pondrá en marcha el componente *Estadísticas*.

7.5 Diagrama de despliegue

A continuación se muestra el diagrama de despliegue de los componentes explicados en los apartados anteriores.

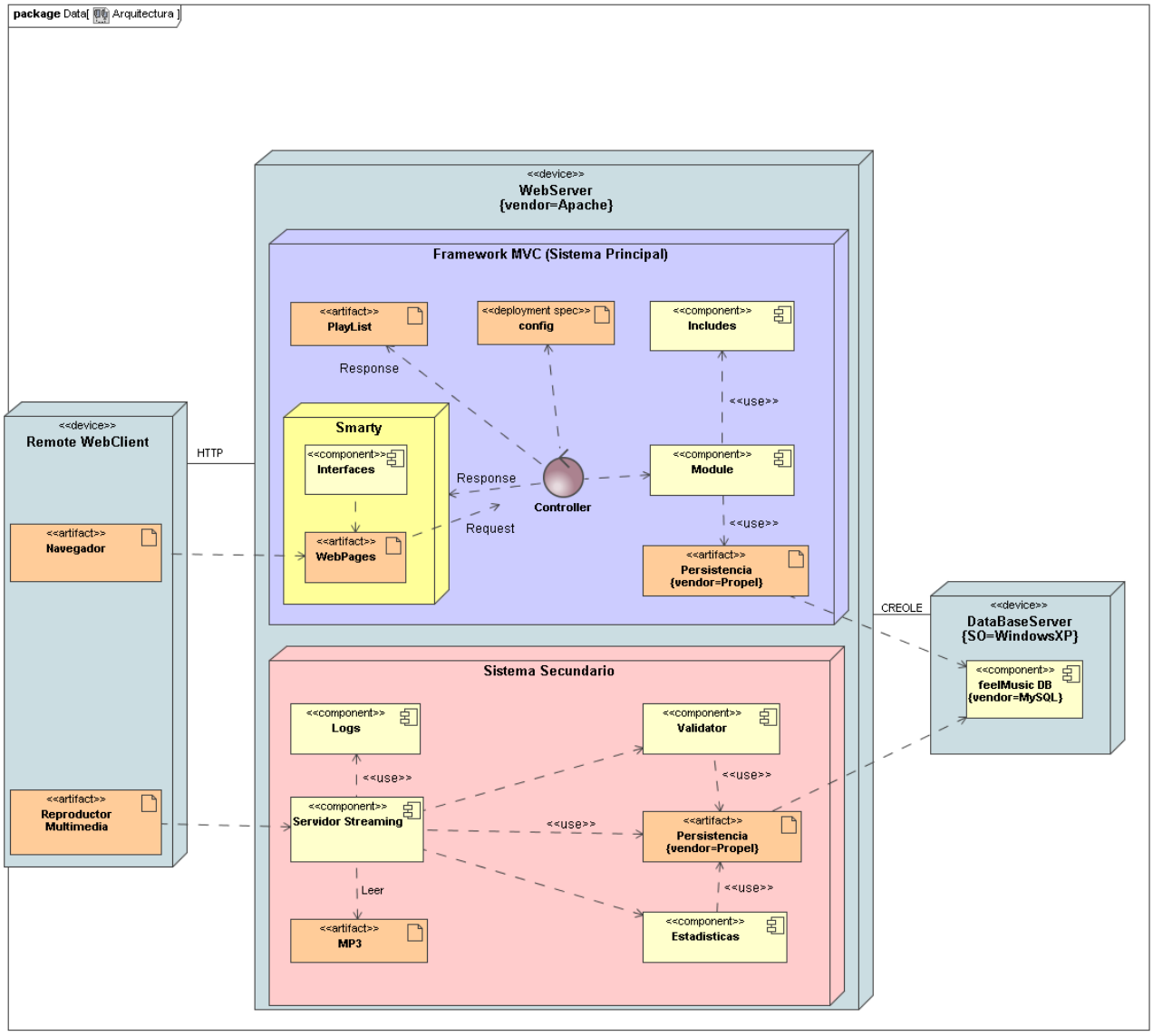


FIG 56: Arquitectura del sistema

7.6 Especificación del entorno tecnológico

Antes de especificar el entorno tecnológico del sistema se recuerda que la aplicación ha sido diseñada para un uso personal. Al tratarse de un uso personal se espera instalar la aplicación en un Pc de sobremesa y con una conexión normal a Internet, hasta 1Mb de bajada y 300 kbps de subida.

Condiciones que ha de reunir el Servidor

Para el funcionamiento de la aplicación es necesario disponer del Servidor Apache, versión 2 (más información capítulo 6.3) y la base de datos MySQL, versión 5 (más información capítulo 6.2). Estos dos elementos podrán ser instalados en el proceso de instalación.

Las características del PC para un rendimiento óptimo son: sistema operativo Windows XP, versión Home o Profesional, un procesador a 2000 Mhz y 512 MB de memoria RAM.

El acceso al servidor podrá realizarse a través de una red local o Internet.

Se han realizado pruebas sobre un red local con excelentes resultados. En cambio cuando la conexión se realizada mediante Internet el rendimiento disminuye,obviamente, ya que se utilizó una conexión comercial en la que la velocidad de subida no era superior a 300 kbps.

Condiciones del cliente:

Los usuarios que utilicen la aplicación necesitarán los siguientes componentes:

- Módem de conexión a Internet con velocidad mínima 28.800 bps o superior.
- Un navegador de Internet.
- Un reproductor de audio multimedia que soporte la recepción de datos mediante streaming.

7.7 Patrones de diseño

En el siguiente apartado se describen los patrones de diseño utilizados para resolver problemas encontrados en el proceso de desarrollo. Se utilizan los tipos de patrones: GRASP (General Responsability Assignaments Software Patterns), encargados de asignar responsabilidades y GoF los cuales indican como se crean, se componen o representan los objetos de manera independiente. Además de estos dos tipos de patrones se utilizan patrones que describen como crear frameworks (marcos de ejecución, conjunto de clases que dan el entorno de ejecución a una aplicación)

7.7.1 Patrones GRASP

Expert

Este patrón resuelve el problema de asignación de responsabilidades para los objetos. El patrón propone asignar la responsabilidad al objeto experto en información, es decir la clase que dispone de la información necesaria para realizar la responsabilidad. Utilizando este patrón se consigue un sistema sencillo, fácil de entender, mantener, ampliar y la posibilidad de reutilizar componentes en futuras aplicaciones.

La clase Directorio que permite analizar un directorio del sistema de archivos es un ejemplo de aplicación.

Creator

El patrón creator resuelve el problema de asignar responsabilidades cuando se crea una nueva instancia de una clase. Creator propone la siguiente solución al problema:

Se asigna a la clase B la responsabilidad de crear una instancia de la classe A, en el caso que B agregue, contenga, registre, utilice o tenga datos de inicialización del objeto de A.

Por ejemplo los controladores de los módulos sencillos se encargan de crear los objetos del modelo de negocio.

Low Coupling

Este patrón es uno de los patrones que solventa el problema del acoplamiento. El acoplamiento básicamente es la dependencia masiva de un elemento respecto a otros elementos. Si una sistema tiene un alto acoplamiento implica tener los siguientes problemas:

- Los cambios realizados en las clases relacionadas obligan a realizar cambios locales.
- Clases difíciles de entender.
- Dificultades para la reutilización de código.

La agrupación de las clases en función de sus funcionalidades es un ejemplo de Low Coupling.

High Cohesion

Este patrón esta relacionado con el anterior. Permite resolver el problema de gestión de complejidad. Fundamentalmente el problema se encuentra en que una clase con una baja cohesión realiza muchas tareas que no le son relacionadas, con lo cual se hace difícil de entender, mantener, y reutilizar.

La mayoría de clases del sistema tiene un diseño con un alto nivel de cohesión, por ejemplo la clase Logs se encarga únicamente de escribir un fichero de logs.

Controller (controlador)

La utilización de este patrón soluciona el problema de como se debe asignar la responsabilidad para tratar las entradas al sistema. En otras palabras, el patrón controller permite crear una o diversas clases encargadas de controlar el flujo de dialogo del sistema.

El controlador del framework MVC es un claro ejemplo de controller. No tan sólo se utiliza este controlador, cada módulo sencillo tiene el suyo.

Polimorfismo

El patrón polimorfismo soluciona el problema de tratamiento de procesos alternativos basados en la tipología de los parámetros recibidos. La utilización de este patrón permite mantener y añadir nuevas tipologías de clases con cierta facilidad. Existe un inconveniente, cuando se añade una nueva tipología se debe modificar el código de la clase que instancia los tipos de clases tipologícas. Para solucionar este problema se aplica el patrón Factory Method, más información en el siguiente apartado.

El patrón polimorfismo es aplicado en el proceso de lectura de metados de los ficheros de audio (ver capítulo 6.11) . El framework MVC también utiliza este patrón para definir la tipología de interficie de datos.

7.7.2 Patrones GoF

Factory Method

Este patrón soluciona el problema de creación de objetos cuando se tiene una o más de una clase tipologica (Patrón Polimorfismo) que implementa una interficie. La solución consiste en crear una clase que en función del parámetro de entrada instancia una clase de las que implementa la interficie.

Factory Method complementa la solución propuesta para la lectura de metadatos de los ficheros de audio (ver capítulo 6.11). El framework MVC lo utiliza para crear el tipo de objetos interficie.

Facade

Este patrón facilita el uso de subsistemas que se quieren incorporar a un sistema principal. Además minimiza la dependencia del segundo susbistema, con lo cual si se producen migraciones del subsitema las consecuencias sobre el sistema principal son inapreciables.

La clase ScanFile encargada de leer los metadatos de los ficheros de audio utilizando el

subsistema GetID3 (ver capítulo 6.11) es un ejemplo de Facade.

7.7.3 Frameworks

Model View Controller (MVC)

El patrón MVC define un conjunto de clases que permiten crear una arquitectura de tres capas:

- El *Model,* Objetos de la capa de negocio.
- La View, Objetos de visualización.
- El Controller que reacciona a las entradas del sistema y refleja la capa de negocio

  En el capítulo 7.5 y capítulo 6.9 se explica con mayor profundidad como se aplica este patrón.

Data Access Objects (DAO)

Este patrón propone una solución al problema de la persistencia de objetos en base de datos relacional. La solución consiste en modelar objetos del modelo de negocio los cuales no necesitan conocer el destino final de la información que esta manipulando. Estos objetos han de tener como mínimo las siguientes operaciones: recuperar llave primaria, crear, modificar y borrar.

Al utilizar el patrón DAO se crea el efecto de tener una base de datos de objetos virtual.

DAO se aplica utilizando el sistema de persistencia Propel, como ya se ha dicho en apartados anteriores.

8 Conclusiones

Llegados a este punto con el proyecto finalizado. Podemos decir que el resultado ha sido altamente satisfactorio. Se han cumplido tanto los objetivos principales del proyecto como los secundarios que fueron planteados inicialmente.

Referente a los objetivos principales se han conseguido crear dos nuevos métodos de búsqueda que pueden calificarse como naturales e intuitivos. Si se utiliza la búsqueda en base a las observaciones personales los usuarios consiguen localizar la música que desean escuchar sin tener que recordar títulos de canción o álbum, solamente necesitan el nombre de un amigo o lugar. Respecto a la búsqueda por "emociones personales" los usuarios consiguen listados de música basados en la caracterización dada a cada "emoción personal" o tag. Los listados se consiguen con tan sólo un clic lo cual agiliza la búsqueda de música.

La posibilidad que tienen los usuario de crear un listado con canciones favoritas, la creación de playlist personalizados y la agenda, encargada de recordar observaciones personales, son parte de los objetivos secundarios que al igual que los objetivos principales facilitan la localización de música.

Para facilitar la navegación de la aplicación se han utilizado iconos. En mi opinión estos ayudan notablemente al uso de la misma ya que permiten identificar las características y elementos de la aplicación de manera intuitiva y directa. No sólo los iconos sino los mensajes que aparecen al colocar el ratón encima de los diferentes elementos que componen las interficies ayudan al usuario a identificarlos y sentirse más cómodo.

La experiencia de incorporar los servicios de Google en la aplicación ha sido exitosa, ya que los usuarios pueden utilizarlos para recuperar enlaces con información/noticias de artistas, álbumes, ... sin tener que abandonar la aplicación.

Aunque se han conseguido los objetivos del proyecto soy consciente de que hay algunos puntos que necesitarían mejorarse. Por ejemplo incrementar las seguridad de la aplicación, mejorar el rendimiento del algoritmo de búsqueda de ficheros MP3 o augmentar las prestaciones de Streaming. Estas mejoras podrían formar parte de proyectos futuros.

Referente al lenguaje de programación estoy muy satisfecho de haber elegido PHP, ya que he conseguido entender el funcionamiento de un lenguaje desconocido antes del inicio del proyecto. Además de aprender un lenguaje nuevo e aprendido todas las tecnologías que lo envuelven y lo soportan con lo cual me convierten en un buen profesional del lenguaje PHP. En mi opinión PHP seguirá teniendo el mismo éxito que tiene actualmente con el paso del tiempo ya que su dinamismo y agilidad lo convierten en el lenguaje ideal para el desarrollo de aplicaciones web de tamaño pequeño y mediano.

No solamente se han cumplido los objetivos del proyecto sino que los objetivos personales han sido conseguidos también con un alto grado de satisfacción personal.

Sobretodo se quiere destacar la cantidad de conceptos nuevos y tecnologías que se han aprendido. Como son: el Streaming, la composición de los ficheros MP3, la creación de ficheros XML y la configuración y puesta en marcha de un servidor web.

Durante el proceso de estudio de conceptos y tecnologías se ha mejorado la capacidad de aprendizaje, investigación y especialmente extraer conclusiones y tomar decisiones.

Además de aprender se han aplicado conceptos estudiados en las materias de Ingeniería del software, Proyectos Informáticos, Bases de Datos, Nuevas tecnologías de los sistemas de información y Redes.

Haciendo referencia a la asignatura Ingeniería del software, personalmente había especial interés en aplicar y ampliar los conocimientos estudiados, especialmente el tema de metodologías y utilización de patrones. Con el proyecto concluido puedo decir que mis conocimientos de ingeniería del software se han ampliado y mejorado. Además se ha despertado en mi un especial interés y entusiasmos por esta rama de la informática.

Al no tener un cliente y ser yo el autor e iniciador del proyecto me ha ofrecido la oportunidad de caracterizar el mismo. Este proceso de caracterización me ha permitido entender que es muy importante definir los límites de un proyecto desde su nacimiento, sobretodo si se quiere conseguir un proyecto de calidad.

Durante el desarrollo del proyecto se han estudiado tanto herramientas de trabajo como tecnologías disponibles para diferentes problemas. Este proceso ha hecho mejorar mi capacidad de análisis y toma de decisiones. Pienso que esta habilidad puede ser importante cuando forme parte de un equipo de trabajo y haya de analizar y exponer mis opiniones a los miembros del equipo.

Por último decir que todo lo aprendido, experimentado y vivido será de utilidad en mi carrera profesional y personal.

Bibliografía

- Mike O'Docherty (2005); Object-Oriented Analysis and Design, Underestanding system development with UML 2.0. England, John Wiley & Sons, Ltd.
- Alberto Manuel Rodrigues da Silva, Carlos Alberto Escaleria Violeira (2001); UML: Metodologias e ferramentas CASE. Portugal, Ediciones Centro Atlantico.
- Craig Larman (2002) traducción: Begoña Moros Valle. UML y patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado. Madrid: Prentice Hall.
- Josep Bosch i Andreu (2003); UML, Disseny de software orientat a objectes i Patrons. Girona: Apunts UDG.

- http://www.wikipedia.org
- http://www.ampache.org
- http://www.php.net
- http://pear.php.net
- http://www.service-architecture.com
- http://www.agilemodeling.com
- http://www.onlamp.com
- http://www.wordreference.com
- http://www.rae.es/
- http://propel.phpdb.org/trac/
- http://www.meta-language.net
- http://www.phpobjectgenerator.com
- http://code.google.com/apis/soapsearch/index.html
- http://www.mp3-tech.org
- http://www.id3.org

Anexos

1 MP3

MPEG-1 Audio Layer 3, más conocido como MP3, es un formato de audio digital comprimido con pérdida desarrollado por el Moving Picture Experts Group (MPEG) para formar parte de la versión 1 (y posteriormente ampliado en la versión 2) del formato de video MPEG. Su nombre es el acrónimo de MPEG-1 Audio Layer 3.

1.1 Historia

Este formato fue trabajado principalmente por Karlheinz Brandenburg, director de tecnologías de medios electrónicos del Instituto Fraunhofer IIS, perteneciente a una red de 47 centros de investigación alemanes que junto con Thomson Multimedia controla el grueso de las patentes relacionadas con el MP3.

El formato MP3 se convirtió en el estándar utilizado para *streaming* de audio y compresión de audio de alta calidad gracias a la posibilidad de ajustar la calidad de la compresión, proporcional al tamaño por segundo (bitrate), y por tanto el tamaño final del archivo, que podía llegar a ocupar 12 e incluso 15 veces menos que el archivo original sin comprimir.

Fue el primer formato de compresión de audio popularizado gracias a Internet, ya que hizo posible el intercambio de ficheros musicales.

Tras el desarrollo de reproductores autónomos, portátiles o integrados en cadenas musicales (estéreos), el formato MP3 llega más allá del mundo de la informática.

1.2 Detalles Técnicos

En MPEG-3 (MP3) existen varias diferencias respecto a los estándares MPEG-1 y MPEG-2, entre las que se encuentra el llamado banco de filtros híbrido que hace que su diseño tenga mayor complejidad. Esta mejora de la resolución frecuencial empeora la resolución temporal introduciendo problemas de pre-eco que son predecidos y corregidos. Además, permite calidad de audio en tasas tan bajas como 64Kbps.

El **banco de filtros** utilizado en esta capa es el llamado banco de filtros híbrido polifase/MDCT. Se encarga de realizar el mapeado del dominio del tiempo al de la frecuencia tanto para el codificador como para los filtros de reconstrucción del decodificador.

La **compresión** se basa en la eliminación de información perceptualmente irrelevante, es decir, en la incapacidad del sistema auditivo para detectar los errores de cuantificación en condiciones de enmascaramiento. Este estándar divide la señal en bandas de frecuencia que se aproximan a las bandas críticas, y luego cuantifica cada subbanda en función del umbral de detección del ruido dentro de esa banda. El modelo psicoacústico analiza la señal de audio y calcula la cantidad de ruido que se puede introducir en función de la frecuencia, es decir, calcula la “cantidad de enmascaramiento” o umbral de

enmascaramiento en función de la frecuencia. El codificador usa esta información para decidir la mejor manera de gastar los bits disponibles.El codificador usa esta información para decidir la mejor manera de gastar los bits disponibles.

El **empaquetado o formateador de bitstream** es un bloque que toma las muestras cuantizadas del banco de filtros, junto a los datos de asignación de bits/ruido y almacena el audio codificado y algunos datos adicionales en las tramas. Cada trama contiene información de 1152 muestras de audio y consiste de un encabezado, de los datos de audio junto con el chequeo de errores mediante CRC y de los datos auxiliares. Las tramas empiezan con la misma cabecera de sincronización y diferenciación y su longitud puede variar. Además de tratar con esta información, también incluye la codificación Huffman de longitud variable.

2 Tags ID3

2.1 Historia

Los tags ID3 surgen con posterioridad al estándar MP3. Eric Kemp, un usuario anónimo, señaló la necesidad de catalogar los ficheros de sonido con información textual básica de su procedencia: autor, título, etc. Así, sugirió la posibilidad de incluir dichos *metadatos* al final de cada fichero MP3. Eventualmente, esta idea fue implementada con gran éxito entre los usuarios, naciendo la primera versión de ID3. Posteriormente, Michael Mutschler, creador de MP3ext, sugirió la versión 1.1 de ID3. A pesar de su éxito, existían quejas sobre algunas limitaciones técnicas del formato de las etiquetas. Por ello, se elaboró la versión 2 de este estándar informal.

2.2 Estructura

Versión 1.0

Consiste en adjuntar un bloque de tamaño fijo de 128 bytes al final del fichero en cuestión. Este bloque contiene las siguientes etiquetas:

- Una cabecera que identifica la presencia del bloque ID3 y su versión, caracteres TAG.
- Título: 30 caracteres.
- Artista: 30 caracteres.
- Álbum: 30 caracteres.
- Año: 4 caracteres.
- Un comentario: 30 caracteres.
- Género (musical): un carácter.

Todas las etiquetas usan exactamente 30 caracteres y caracteres ASCII, excepto el género, que es únicamente un número de índice. El género musical asociado a cada índice está predefinido en el estándar.

Versión 1.1

Un inconveniente de la versión anterior es que no es posible indicar el número de pista correspondiente al *álbum* al que pertenece la grabación. La versión 1.1 simplemente "resta" los dos últimos caracteres de la etiqueta *comentario* para este propósito.

Versión 2.0

Aunque ID3 versión 1.x es suficiente en muchos casos, presenta algunos problemas serios:

- La longitud de las etiquetas es insuficiente para algunas grabaciones.
- El uso de caracteres ASCII impide su uso con lenguas no occidentales.
- El conjunto de etiquetas es insuficiente. Ejemplo, en algunas grabaciones es necesario distinguir el autor del intérprete.
- No es posible incluir nuevas etiquetas no predefinidas, en función de las necesidades de cada usuario. Por ejemplo, las preferencias de ecualización.

Por este motivo surge la versión 2 de ID3. Los detalles técnicos son más complejos que en las versiones anteriores. Las diferencias más significativas son las siguientes:

- Utiliza caracteres Unicode.
- Las etiquetas se sitúan al principio del fichero, no al final. Esto facilita la difusión por Internet mediante streaming, ya que no hay que esperar a que se descargue todo el fichero para conocer las etiquetas.
- Las etiquetas pueden tener mayor o menor longitud. No hay restricciones.
- Es posible incluir imágenes, no solo texto. Por ejemplo, la caratula del álbum.
- Admite etiquetas definidas por el usuario.
- Se han predefinido más de 35 etiquetas estándar.
- La letra de la canción se puede alamacenar bajo el frame Lyrics3 en la TagID3, al igual que la portada del albúm.
- Las etiquetas pueden ser cifradas.

3 PlayList

El termino playlist representa una lista de canciones en su forma más sencilla. Como particularidad el termino playlist tiene más de un significado, dependiendo del campo de utilización, ya sea en las emisiones de radio o para los reproductores de audio en los PC.

3.1 Historia

El termino nació cuando las emisoras de radio empezaron a emitir en formato top 40, las 40 canciones más populares. Entonces las emisoras querían distribuir, publicar y emitir una lista de canciones limitada. El termino playlist identificaba la lista de 40 canciones que una emisora de radio daba a conocer. Cada emisora tenia su formato personal. Además de identificar la lista de emisoras, el termino fue utilizado para referirse a una lista ordenada de canciones que era reproducida durante un periodo de tiempo.

Cuando la música empezó a almacenarse y reproducirse en los ordenadores personales el termino playlist se volvió más popular. Este fue adoptado por varios programas que reproducían e intentaban organizar y controlar la música almacenada en un PC. Tales playlist podían ser definidos, almacenados y seleccionados para ser reproducidos en secuencia o al azar siempre y cuando el programa permitiera reproducción al azar.

La ventaja de estos playlist es que permite incluir tantas canciones como se desee, con ello se puede programar sesiones musicales con diferentes estilos musicales sin que el usuario tenga que interaccionar con el programa reproductor.

Hoy día el termino playlist representa la lista de canciones o archivo de audio que puede ser creado para organizar librerías musicales. Existen diferentes tipos de ficheros playlist que permiten organizar las librerías. Estos son m3u, pls, asx, smil, kpl y XSPF. El más popular y utilizado es el m3u.

3.2 M3U

M3U, es el acrónimo de MPEG versión 3.0 URL. M3U es un tipo de fichero que guarda información multimedia de los playlist. El creador de este tipo de fichero fue la empresa Winamp, aunque actualmente este fichero ha sido adoptado por muchas aplicaciones, como por ejemplo XMMS, RealPlayer, Windows Media Player o iTunes.

Estructura

Es un fichero de texto plano que contiene la localización de uno o más ficheros de audio que pueden ser reproducidos por un reproductor. Cada localización es una línea nueva en el fichero. La localización puede ser un ruta absoluta o relativa (Ex:"C:\Musica\Fama.p3" o "Fama.mp3") o incluso puede ser una URL (http://localhost/feelMusic/modules/streaming/play.php?song=303). El fichero además permite incluir comentarios, para incluirlos se utiliza el carácter #.

Existe una segunda versión de este tipo de fichero, se trata de la versión *extendida.* Esta versión mejorada permite incluir directivas para informar al reproductor de la duración de la canción, el artista y

el título de la canción. La directiva que informa de esta información es #EXTINF. Se sitúa en la línea posterior de la localización del fichero.

El uso más común de M3U es la creación de playlist que permite el acceso a datos en Internet, como por ejemplo la conexión con emisoras de radio o la descarga de ficheros de audio.

La extensión de los ficheros es M3U.

Ejemplo estructura M3U

Queen.m3u

```
#EXTM3U //Directiva informa de la versión extendida
#EXTINF:5:42,Queen - Great king rat.mp3 //Directiva de información audio
http://localhost/feelMusic/modules/streaming/play.php?song=779 //Localización
#EXTINF:4:23,Queen - The Night Comes Down.mp3
http://localhost/feelMusic/modules/streaming/play.php?song=780
#EXTINF:3:44,Queen - Jesus.mp3
http://localhost/feelMusic/modules/streaming/play.php?song=782
```

4 Streaming

Streaming es un término que describe una estrategia sobre demanda para la distribución de contenido multimedia a través del internet.

4.1 Historia

Antes de que la primera instancia de tecnología streaming apareciera en abril de 1995 (con el lanzamiento de RealAudio 1.0), la reproducción de contenido multimedia mediante internet necesariamente implicaba tener que descargar completamente el "archivo contenedor" al disco duro local. Como los archivos de audio —y especialmente los de video— tienden a ser enormes, su descarga y acceso como paquetes completos se vuelve una operación muy lenta.

Sin embargo, con la tecnología del streaming un archivo puede ser descargado y reproducido al mismo tiempo, con lo que el tiempo de espera es mínimo.

4.2 Estructrua

Para poder proporcionar un acceso claro, continuo y sin interrupciones, el streaming se apoya en las siguientes tecnologías:

Códecs

Codec es una abreviatura de Codificador-Descodificador. Describe una especificación implementada en software, hardware o una combinación de ambos, capaz de transformar un archivo con un flujo de datos (stream) o una señal. Los códecs pueden codificar el flujo o la señal (a menudo para la transmisión, el almacenaje o el cifrado) y recuperarlo o descifrarlo del mismo modo para la reproducción o la manipulación en un formato más apropiado para estas operaciones. Los códecs son usados a menudo en videoconferencias y emisiones de medios de comunicación.

Protocolos Ligeros

UDP y RTSP (los protocolos empleados por algunas implementaciones de streaming) desempeñan entregas de paquetes de servidor-a-cliente con una velocidad mucho mayor a la que se obtiene por TCP y HTTP. Esta eficiencia es alcanzada por una modalidad que favorece el flujo continuo de paquetes. Cuando TCP y HTTP sufren un error de transmisión, siguen intentando transmitir los paquetes perdidos hasta conseguir confirmación de que la información llegó en su totalidad; sin embargo, UDP continúa mandando los paquetes sin tomar en cuenta interrupciones, ya que en una aplicación multimedia estas pérdidas son casi imperceptibles.

Precarga

Las entregas de paquetes de servidor-a-cliente pueden estar sujetas a demoras conocidas como lag, un fenómeno ocasionado cuando los paquetes escasean (debido a interrupciones en conectividad o sobrerrecarga en el ancho de banda). Por lo tanto, los reproductores multimedia precargan, o almacenan en el buffer, los datos que van recibiendo para así disponer de una reserva de contenido destinada a

reproducirse durante un lag.

Red de Distribución de Contenido

Si un determinado stream comienza a atraer una cantidad de usuarios mayor a su capacidad de ancho de banda, estos usuarios comenzarán a experimentar lag. Eventualmente llega un punto en que la calidad del stream es peor de lo que un usuario normal puede tolerar. Ofreciendo soluciones, surgen empresas y organizaciones que se encargan de proveer ancho de banda exclusivamente para streaming, y de apoyar y desarrollar estos servicios.

4.3 Usos

Los principales usos de streaming son:

- Radio por Internet
- Interlacing
- Televisión por Internet
- Virtual Network Computing

4.4 Propiedad Intelectual

Para no tener problemas de propiedad intelectual la mayoría de las implementaciones del streaming han sido concebidas y diseñadas para desechar los datos recién interpretados.

5 Freedb

freedb es una base de datos que almacena información sobre la lista de canciones de los compact discs (CDs). Todo el contenido de la base de datos esta protegido por la licencia GNU (General Public License). Su creación tomo como modelo la base de datos CDDB (compact disc database). CDDB en sus inicios era de uso público pero actualmente tiene un uso comercial.

El 24 Abril del 2006 la base de datos tenía registrados cerca de 2.000.000 CDs. Para buscar la información de los CD utilizando Internet es necesario un programa cliente que calcula un identificador (ID) único. Una vez se tiene el ID el programa cliente hace la petición a freedb. Si el disco esta en la base de datos el cliente podrá recibir y informar al usuario del artista, titúlo del álbum, número de pista y información extra del CD que esta escuchando.

Freedb es utilizado por programas reproductores de audio, programas catalogadores y programas CD ripper.

6 Problema Con Freedb

Antes de exponer el problema que ha impedido utilizar freedb (ver anexo 5) se presenta el uso que se quería dar a freedb en el proyecto. Una vez conocido el uso se introduce el requisito de acceso a freedb. Por último se describre el problema y sus posibles soluciones.

6.1 ¿Porque Utilizar Freedb?

Como ya se ha descrito en las características generales (capítulo 3), el objetivo del proyecto es organizar los ficheros de audio MP3 (más información anexo 1). Para organizarlos se utiliza la información musical, guardada en los tags ID3(más información anexo 2), que contiene los MP3. En algunos casos la información se encuentra incompleta o sin normalizar. Para completar y normalizar la información se pretendía utilizar freedb.

6.2 Requerimiento De Acceso A Freedb

Para poder acceder a la información de un álbum almacenado en freedb se requiere un identificador único (disco ID).

¿Que información se necesita para generar el disco ID?

Información necesaria para calcular el disco ID:

- Datos almacenados en la *tabla de contenidos* del CD original
- Número de canciones que contiene el CD
- Algoritmo de cálculo

Los datos de la *tabla de contenido* son: minutos, segundos y frame offset (posición de inicio de cada canción).

El algoritmo de calculo del disco ID es proporcionado por freedb.

Con esta información es posible generar un disco ID pero si el disco ID no es generado correctamente no será compatible con freedb y por lo tanto no se obtendrán los datos esperados.

6.3 Problema

La información obtenida de los ficheros MP3 no es suficiente para calcular el disco ID.

Posibles soluciones

1. **Implementar una solución para obtener información de los cd originales (cd ripper)**

Esta solución no fue planteada en los requerimientos iniciales. Plantear el desarrollo de esta solución no es posible porque no forma parte del objetivo del proyecto. Recuerdo que el objetivo es desarrollar una aplicación que permite organizar y gestionar la música almacenada en un pc

convencional.

2. Base de datos alternativa

La alternativa a freedb es Gracenote (www.gracenote.com). Gracenote ofrece un servicio comercial por el cual se ha de pagar para utilizarlo.

Dado el ámbito académico del proyecto no se plantea utilizar un servicio comercial.

3. Obtener el disco ID del tag id3v2

El tag id3v2 (más información anexo 2) permiten guardar el disco ID. Aunque pueda almacenarse el disco ID, no se puede asegurar que se encuentre siempre. Se ha echo un estudio sobre el número de apariciones del disco ID, el resultado ha sido el siguiente:

Número de apariciones del disco ID

- Número de archivos MP3 analizados: 2619
- Número de apariciones del disco ID: 5

  La aparición del disco ID ocurre en un 0'19% de los casos.

Dado el resultado del estudio se descarta la posibilidad de desarrollar un módulo que obtenga el disco ID de los tags ID3. La productividad del módulo sería casi nula.

6.4 Consecuencias De No Utilizar Freedb

1. Los ficheros MP3 que no tienen toda la información quedaran incompletos en la base de datos.
2. No se podrá normalizar los datos guardados en la base de datos.

Para solucionar la primera consecuencia se han definido nombres genéricos que servirán para nombrar la información incompleta.

Información Incompleta	***Nombre***
Álbum	ORPHANED
Genero	UNKNOWN
Artista	ANONIME
Nombre	Nombre archivo

Table 1: Información archivos incompletos

La segunda consecuencia no tiene solución.

6.5 Conclusiones

Una vez explicado el objetivo de freedb en el proyecto, expuesto el problema de utilización y sus posibles soluciones, e por último se han estudiado las consecuencias se ha llegado a la conclusión que no será desarrollado el módulo de comunicación con freedb.

7 Tecnologías Propel

Propel esta basado en Apache Torque, un buen ORM para aplicaciones Java.

Propel ofrece dos herramientas que trabajan juntas para obtener una persistencia de objetos eficiente según las especificaciones de cada SGBD: *Generator* y *Runtime Framework*.

7.1 Herramientas

Generador

Su funcionamiento esta basado en un simple fichero XML que describe el esquema de tú base de datos y las tablas. El generador construirá las clases PHP para interactuar con el modelo de datos y un fichero SQL para crear las tablas, las llaves, secuencias etc. El fichero es especifico para cada SGBD. El uso de un sistema de construcción para crear SQL específicos según el SGBD hace que la base de datos se realmente independiente del sistema que se esta construyendo sin sacrificar su rendimiento.

Runtime

Runtime Framework permite utilizar las clases generadas en los scripts de PHP de tal manera que actúan transparentemente en la lectura y escritura de la base de datos.

7.2 Tecnologías Relacionadas

Phing

Propel utiliza Phing v2 (http://phing.info) para realizar la construcción de operaciones. Phing es un framework construido en PHP5 basado en Apache Ant. Propel ha ampliado las funcionalidades básicas de Phing para realizar tareas especificas para Propel. Han creado un generador de objetos para las aplicaciones PHP.

Creole

Creole (http://creole.phpdb.org) es un sub-projecto de Phing, el cual proporciona una API uniforme para la conexión con la base de datos (abstracción de base de datos). La API es especificar para PHP5.

Creole se basa en la API JDBC (Java Database Conectivity). Tiene muchas ventajas: incluye una API orientada a objetos, unificación de diferentes tipos de sistemas, soporte de metadatos y un framewrok pare realizar pruebas de test y poder asegurar el buen comportamiento de los drivers. Los drivers soportados son: Oracle, MySQL, Microsoft SQL Server y PostresSQL.

PEAR::Log

Actualmente es el único modulo PEAR necesario por Propel. El package Log permite un camino sencillo y flexible para gestionar logs.

8 PEAR

Es un framework y sistema de distribución de componentes reusables de PHP.

Para el proyecto han sido necesario los package:

- *Log*: Requerido por Propel.
- *Validate*: Requerido por el Framework MVC.

Mas información: http://pear.php.net/

9 Aplicaciones Estudiadas

9.1 Ampache

Características generales

Nombre	Ampache
Versión	3.3.2 - Beta 1
Web	www.ampache.org
En desarrollo	Si
Licencia	GPL
Tiempo instalación	6 horas
Tiempo adaptación	3:30 horas

Requisitos de instalación

Lenguaje	PHP 4 o 5
Servidor	cualquiera (Apache)
Base de datos	MySQL, 3.23 y 5.0.17
Otros	- gd - iconv.dll - ZLib - 16 Mb RAM

Organización de los ficheros de audio

Utilizando estructura directorios maquina cliente	No
Información tag ID3v1	Si
Información tag ID3v2	Si
Tipo de ficheros reconocido	mp3, mp2, ogg, wav, wma, flac

Información almacenada

Base de datos	Si
Utilizar ficheros	No

Búsquedas

- Título canción, álbum, Artista, Género, Nombre del archivo,

  Criterios posibles: Reproducida (Si o No), Mínimo bitrate y máximo resultados

Otras características técnicas

Streaming	Si
Acceso remoto (Internet)	Si, Diferentes niveles usuarios con privilegios.
Listas de reproducción	- Formatos: simple m3u, extensa m3u, pls, asx, ram. - Acceso público o privado resto usuarios.
Editor tags ID3	No, Actualiza si han estado modificados por una aplicación externa.
Información base datos publica	No
Agenda musical	No
Última fecha de reproducción	No
Últimas canciones escuchadas	Si
Clasificación personal	No
Top 10 (Home)	Álbumes, Artistas y Canciones
Información nuevos archivos audio	Si
Busca información / noticias artistas	No
Puntuar canciones	No
Otras	- Compartir música con otros servidores Ampache.

Problemas

- Una vez finalizada la instalación no se tenia acceso al programa, no conseguía pasar la pantalla login.

  Solución: Actualizar la versión de PHP 4 a PHP 5.

- Error PHP, no se encontraba una función necesaria para ampache

  Solución: Actualizar Ampache 3.3.1.6 (versió estable) a 3.3.2 - Beta 1

- Las búsquedas de archivos de audio no funcionan, problemas internos.

Más funcionalidades

- Mostrar en la página de inicio la canción que se esta reproduciendo en la maquina cliente.
- Algunas de las funcionalidades que recibe un ***Artista*** son:
  - Mostrar sus canciones y reproducirlas. Puedes seleccionar que canciones quieres reproducir o realizar un sorteo.
  - Actualizar la información de la base de datos si los tags ID3 han sido modificados.
- Para cada genero informar del número de artistas, álbumes y canciones
- Un usuario puede enviar emails a otros usuarios o al administrador.
- Apartado de configuración para las preferencias del usuario. Configuraciones visuales y variables

del sistema como el tiempo de logout.

- Las canciones que no es possible conocer su álbum son añadidas a uno común llamado *orphaned.*
- Si una canción no quiere ser encontrada puede ser deshabilitada. No se borra del sistema simplemente es ignorada por el buscador.
- La música es clasificada por catálogos que el usuario puede administrar. Algunas de las operaciones disponibles para los catálogos son:
    - Mantenimiento (alta, baja y modificación).
    - Buscar canciones repetidas o las canciones deshabilitadas.
    - Listar las canciones marcadas para revisar por parte de los usuarios.
    - Borrar la información de las estadísticas del TOP 10.
- Buscar caratulas en la pagina web de amazon.

Funcionalidades de Seguridad

- Soporta SSL
- No permite usuarios con password en blanco
- Las sesiones terminan automáticamente después de x tiempos sin realizar peticiones al servidor. X puede ser configurado por el usuario administrador.
- Limita el rango de IP's que acceden al servidor.
- Diferentes niveles de acceso.
- El tiempo de vida de un playlist es el mismo que el de una session. Cuando se expira el tiempo de una session o cuando usuario la abandona el playlist queda inoperativo.

Conclusiones

Las característica de esta aplicación se ajustan bastante al tipo de proyecto que se quiere desarrollar.

La funcionalidad Streaming es robusta, eficiente y segura, por ello será utilizada como base del proyecto.

El concepto de añadir las canciones en un álbum será utilizado en el proyecto.

En esta aplicación se ha descubierto un modulo de php que permite la manipulación de los tags ID3. Este módulo se llama getID3, más información en www.getid3.org

Las opciones de seguridad són muy interesantes, especialmente el control de sessions sobre los playlist y la possibilidad de limitar el rango de IP's que tienen acceso al servidor. Pese a esto no se va ha poder implementar estas funcionalidads ya que se salen del marco del proyecto.

9.2 Kplaylist

Características generales

Nombre	kplaylist
Versión	1.6
Web	www.kplaylist.com
En desarrollo	Si
Licencia	GPL
Tiempo instalación	2 horas
Tiempo adaptación	3 horas

Requisitos de instalación

Lenguaje	PHP 4 o PHP 5
Servidor	cualquiera (Apache 1.3)
Base de datos	MySQL 3.23 , 4.0 y 5.0.7

Organización de los ficheros de audio

Utilizando estructura directorios maquina cliente	La reconoce permite navegar
Información tag ID3v1	Si (ver 2 comentario)
Información tag ID3v2	Si
Tipo de ficheros reconocido	mp3, mp2, ogg, wav, wma, flac

Información almacenada

Base de datos	Si
Utilizar ficheros	Si, accede a los ficheros audio-

Búsquedas

- Título canción, álbum, Artista, Género, Nombre del archivo,
- Seleccionar la búsqueda en los TagsID3 o en los ficheros.

Otras características técnicas

Streaming	Si
Acceso remoto (Internet)	Si, Diferentes niveles usuarios con privilegios.
Listas de reproducción	- Formatos: extensa m3u, asx. - Acceso público o privado resto usuarios. - Creación al azar de playlist. - Administración playlist ágil y intuitiva.
Editor tags ID3	No
Información base datos publica	No
Agenda musical	No
Última fecha de reproducción	No
Últimas canciones escuchadas	Si
Clasificación personal	Los grupos más escuchados.
Top 10 (Home)	Si
Información nuevos archivos audio	
Busca información / noticias artistas	No
Puntuar canciones	

Problemas

- Los playlist no son contruidos correctamente.

Más funcionalidades

- Para reconocer la música que se quiere catalogar, la aplicación dispone de una funcionalidad que permie navegar por el árbol de directorios de la máquina local. Cuando se localiza un archivo a catalogar permite añadirlo. Tambíen es possible definir un directorio base apartir del cual catalogar todos los archivos de audio localizados.
- Permite a los usuarios escoger las preferencias de visualización.
- Enviar emails con listas de reproducción incluidas
- Busca información musical en http://www.last.fm.
- Publicar notícias (boletines) para todos los usuarios.
- Administrar el bitrate de las canciones utilizando la libreria LEAME.

Seguridad

- Soporata SSL
- Las sesiones terminan automáticamente después de x tiempos sin realizar peticiones al servidor. X puede ser configurado por el usuario administrador.

- Diferentes niveles de acceso.
- Crear registro de control de acceso en el servidor.
- El administrador puede bloquear el acceso de los usuarios al sistema.
- El tiempo de vida de un playlist es el mismo que el de una session. Cuando se expira el tiempo de una session o cuando usuario la abandona el playlist queda inoperativo.

Conclusiones

Es una aplicación sencilla, con las necesidaes justas y necesarias para la administación musical. Las características que pueden ser útiles par el proyecto son:

- La creación de las listas de reproducción personalizdas es muy ágil e intuitiva.Se aplicará un sistema semegante en el proyecto.
- La información de los archivos queda en los ficheros no es trasladada a la base de datos. Es una possibilidad interesante para liberar espacion en la base de datos, pero descarta para el proyectoya que hoy en dia las based de datos relacionales estan muy abanzadas y no necesitan restriciones de este tipo.

Uno de los inconvenientes es la disposicion de los menus de la interficie gráfica no son usables. Son un modelo de como NO hay que diseñar los menus. Por lo tanto en el proyecto se intentará no seguir este tipo de modelo.

9.3 Jinzora

Característiques generals

Nombre	JINZORA
Versión	2.3.7
Web	`www.jinzora.org`
En desarrollo	Si
Licencia	GPL
Temps Instal·lació	14 hores - muchos problemas
Temps en familiaritzar-se	03:30:00

Requisitos de instalación

Lenguaje	PHP 4.2
Servidor	Apache
Base de datos	MySQL 4 y 5
Otros	Librerias necessarias:

Lenguaje	PHP 4.2
	- GD library - Iconv.dll
Configuración php.ini	max_execution_time: 300+ memory_limit: 32M+ post_max_size: 32M+ file_uploads: 1 (on) upload_max_filesize: 32M+

Organización de los ficheros de audio

Utilizando estructura directorios maquina cliente	Si
Información tag ID3v1	Si
Información tag ID3v2	Si
Tipo de ficheros reconocido	mp3, mp2, ogg, wav, wma, flac.

Información almacenada

Base de datos	Si
Utilizar ficheros	No

Busquedas

- Título canción, álbum, Artista, Género, Nombre del archivo.

Otras características técnicas

Streaming	Si
Acceso remoto (Internet)	Si
Listas de reproducción	Si, Creación aleatòria y personalitzada
Editor tags ID3	No
Información base datos publica	No
Agenda musical	No
Última fecha de reproducción	Si
Últimas canciones escuchadas	Si
Clasificación personal	No
Top 10 (Home)	Si, personalizado
Información nuevos	Si
Busca información / noticias artistas	No
Puntuar canciones	Si

Streaming	Si
Otras	- organitza vídeos - Connectar equipo música, para reproducir

Problemas

- Despúes de la instalación la dos funcionaliades de catalogar no estaban operativas.
- El servidor de streaming no conseguia emitir información
- La pantalla de login dejó de funcionar despúes de haver sido utilizada otras veces.

Otras funcionalidades

- Permite dos modelos de catalogación. Una siguiendo la estructrua de archivos definida por el usuario en su arbol de directorios. La segunda se realizar a partir de la información obtenida en los tags ID3.
- Si el ordenador esta conectado a un equipo wi-fi, permite realizar la reproducció utilizando el equipo de audio.
- Descarga de Internet información caratulas álbum, fotos artistas, biografias, revistas y letras de canciones.
- Permite la personalización de la interfície y configuracion de las preferencias del sistema.

Seguridad

- Soporta SSL
- No permite nombre de contrasenya y usuarios iguales
- Eliminar un directorio pro

Conclusiones

Aplicación muy compleja, inestable y poco fiable.

No se puede utilizar como patron o guía para el tipo de proyecto que se va a contruir. Demasiadas funcionalidades que no pueden llevarse acabo y no se adecuan al tipo de proyecto que se va a realizar.

9.4 Conclusiones Comunes

En este apartado se hace una valoración del análisis realizado sobre los tres catalogadores (ampache,kplaylist y jinzora).

Ha sido todo un éxtio el análisi, ya que hasta el planteamiento de la idea del proyecto no había tenido ningún contacto con este tipo de aplicaciones. Con el analisis he conseguido entender lo que es y cuales son las funcionalidad que aportan.

El proceso de instalación de los programas ha servido para familiarizarme con el lenguaje PHP, que hasta el momento no se habia tenido ningun contacto, y con la utilización de la base de datos MySQL y la instalación del servidor Apache. al igual que PHP se desconocian.

Por lo que hace referencia a las funcionalides, todas ellas son aplicaciones mas o menos completas que hace años que se estan desarollando. El proyecto que se va ha desarollar se basará en algunos de los rasgos mas característicos de ellas, para ver las características ver capítulo 3 de la documentación.

Todo el tema de seguridad sería muy interesante de implementar pero devido al marco del proyecto (ver capítulo 1) no se van ha llevar acabo las funcionalidades descritas.

10 Google SOAP Search API (Beta)

A continuación se muestra un ejemplo del fichero XML necesario para realizar la petición al servicio doGoogleSearch:

10.1 Petición (Request)

```
<?xml version='1.0' encoding='UTF-8'?>

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
  <ns1:doGoogleSearch xmlns:ns1="urn:GoogleSearch"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <key xsi:type="xsd:string">00000000000000000000000000000000</key>
      <q xsi:type="xsd:string">shrdlu winograd maclisp teletype</q>
      <start xsi:type="xsd:int">0</start>
      <maxResults xsi:type="xsd:int">10</maxResults>
      <filter xsi:type="xsd:boolean">true</filter>
      <restrict xsi:type="xsd:string"></restrict>
      <safeSearch xsi:type="xsd:boolean">false</safeSearch>
      <lr xsi:type="xsd:string"></lr>
      <ie xsi:type="xsd:string">latin1</ie>
      <oe xsi:type="xsd:string">latin1</oe>
    </ns1:doGoogleSearch>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

10.2 Respuesta (Response)

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body>
<ns1:doGoogleSearchResponse xmlns:ns1="urn:GoogleSearch" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<return xsi:type="ns1:GoogleSearchResult">
<documentFiltering xsi:type="xsd:boolean">false</documentFiltering>
<estimatedTotalResultsCount xsi:type="xsd:int">3</estimatedTotalResultsCount>
<directoryCategories xmlns:ns2="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns2:Array" ns2:arrayType="ns1:DirectoryCategory[0]"></directoryCategories>
<searchTime xsi:type="xsd:double">0.194871</searchTime>
```

```
        <resultElements xmlns:ns3="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns3:Array" ns3:arrayType="ns1:ResultElement[3]">
          <item xsi:type="ns1:ResultElement">
            <cachedSize xsi:type="xsd:string">12k</cachedSize>
            <hostName xsi:type="xsd:string"></hostName>
            <snippet xsi:type="xsd:string"> &lt;b&gt;...&lt;/b&gt; on a simple dialog
(via &lt;b&gt;teletype&lt;/b&gt;) with a user, about a &lt;b&gt;...&lt;/b&gt;
http://hci.stanford.edu/&lt;b&gt;winograd&lt;/b&gt;/&lt;b&gt;shrdlu&lt;/b&gt;&lt;br&gt;
. It is written in &lt;b&gt;MacLisp&lt;/b&gt;, vintage 1970, and to
&lt;b&gt;...&lt;/b&gt;</snippet>
            <directoryCategory xsi:type="ns1:DirectoryCategory">
              <specialEncoding xsi:type="xsd:string"></specialEncoding>
<fullViewableName xsi:type="xsd:string"></fullViewableName>
</directoryCategory>
            <relatedInformationPresent
xsi:type="xsd:boolean">true</relatedInformationPresent>
            <directoryTitle xsi:type="xsd:string"></directoryTitle>
            <summary xsi:type="xsd:string"></summary>
            <URL
xsi:type="xsd:string">http://hci.stanford.edu/cs147/examples/shrdlu/</URL>
<title xsi:type="xsd:string">&lt;b&gt;SHRDLU&lt;/b&gt;</title>
          </item>
          <item xsi:type="ns1:ResultElement">
            <cachedSize xsi:type="xsd:string">12k</cachedSize>
            <hostName xsi:type="xsd:string"></hostName>
            <snippet xsi:type="xsd:string"> &lt;b&gt;...&lt;/b&gt; on a simple dialog
(via &lt;b&gt;teletype&lt;/b&gt;) with a user, about a &lt;b&gt;...&lt;/b&gt;
http://hci.stanford.edu/&lt;b&gt;winograd&lt;/b&gt;/&lt;b&gt;shrdlu&lt;/b&gt;/code&lt;br
&gt; . It is written in &lt;b&gt;MacLisp&lt;/b&gt;, vintage 1970, and to
&lt;b&gt;...&lt;/b&gt;</snippet>
            <directoryCategory xsi:type="ns1:DirectoryCategory">
              <specialEncoding xsi:type="xsd:string"></specialEncoding>
<fullViewableName xsi:type="xsd:string"></fullViewableName>
</directoryCategory>
            <relatedInformationPresent xsi:type="xsd:boolean">true
</relatedInformationPresent>
            <directoryTitle xsi:type="xsd:string"></directoryTitle>
           <summary xsi:type="xsd:string"></summary>
            <URL xsi:type="xsd:string">http://hci.stanford.edu/winograd/shrdlu</URL>
<title xsi:type="xsd:string">&lt;b&gt;SHRDLU&lt;/b&gt;</title>
</item>
          <item xsi:type="ns1:ResultElement">
            <cachedSize xsi:type="xsd:string">32k</cachedSize>
            <hostName xsi:type="xsd:string"></hostName>
```

```
            <snippet xsi:type="xsd:string"> &lt;b&gt;...&lt;/b&gt; man and woman through
&lt;b&gt;teletype&lt;/b&gt; and has to &lt;b&gt;...&lt;/b&gt; human diseases) 1970*
Terry &lt;b&gt;Winograd&apos;s&lt;/b&gt; &lt;b&gt;SHRDLU&lt;/b&gt;&lt;br&gt; (Natural
Language Processing &lt;b&gt;...&lt;/b&gt; Lisp Machine Lisp,
&lt;b&gt;MacLisp&lt;/b&gt;, NIL, S-1 &lt;b&gt;...&lt;/b&gt;
</snippet>
            <directoryCategory xsi:type="ns1:DirectoryCategory">
<specialEncoding xsi:type="xsd:string"></specialEncoding>
              <fullViewableName xsi:type="xsd:string"></fullViewableName>
</directoryCategory>
            <relatedInformationPresent xsi:type="xsd:boolean">true
</relatedInformationPresent>
            <directoryTitle xsi:type="xsd:string"></directoryTitle>
            <summary xsi:type="xsd:string"></summary>
            <URL
xsi:type="xsd:string">http://www.trentu.ca/csd/newsarchives/trentu/csp/cr350/79</URL>
            <title xsi:type="xsd:string"></title>
          </item>
        </resultElements>
        <endIndex xsi:type="xsd:int">3</endIndex>
        <searchTips xsi:type="xsd:string"></searchTips>
        <searchComments xsi:type="xsd:string"></searchComments>
        <startIndex xsi:type="xsd:int">1</startIndex>
        <estimateIsExact xsi:type="xsd:boolean">true</estimateIsExact>
<searchQuery xsi:type="xsd:string">shrdlu winograd maclisp teletype</searchQuery>
      </return>
    </ns1:doGoogleSearchResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

10.3 Implementación PHP

```
<html>
<head>
<link rel="stylesheet" type="text/css" href="estils/estil.css">
</head>
<body>
<?php
if(!isset($_POST['clau']))
{
?>
      <form action=<?=$_SERVER['PHP_SELF']?> method="post">
            Elemento a buscar: <input type="text" name="clau"/>
      </form>
```

```
      <?php
}
else
{
$clau = $_POST['clau'];
include("includes/nusoap/nusoap.php");
$soapclient = new soapclient("http://api.google.com/search/beta2");
$params = array(
'key' => '*****************',                  // Google license
key'q'   => $clau,                              // search term
'start' => 0,                                   // start from result n
'maxResults' => 10,                             // show a total of n results
'filter' => false,                              // remove similar results
'restrict' => 'countryES',                              // restrict by topic
'safeSearch' => false,                          // remove adult links
'lr' => '',                                     // restrict by language
'ie' => '',                                     // input encoding
'oe' => ''                                      // output encoding
);
// invoke the method on the server
$result = $soapclient->call("doGoogleSearch", $params,
"urn:GoogleSearch", "urn:GoogleSearch"); // print the results of the search
print_r($result);
?>
<ul >
<?php
foreach($result['resultElements'] as $r)
{
      echo "<li class='resultat'>            //Presentación resultados
                  <span class='titol'>" . $r['title'] . "</span>
                  <br/>" . $r['snippet'] . "<br/>
                  <a href='" . $r['URL'] . "' class='url'>" . $r['URL'] . "
            </a>
            </li>";
}
}
?>
</ul>
</body>
</html>
```

11 Feeds

En los dos siguientes apartados se muestran los formatos XML correspondientes a los *feeds* utilizados, RSS 2.0 y Atom 0.3.

11.1 RSS 2.0

```
<rss version="2.0">
      <channel>
            <generator>NFE/1.0</generator>
            <title>lemurs - Google Noticias</title>
            <link>
                  http://news.google.es/news?ned=es&q=lemurs&ie=UTF-8
            </link>
            <description>lemurs - Google Noticias</description>
            <language>es</language>
            <webMaster>news-feedback@google.com</webMaster>
            <copyright>&copy;2006 Google</copyright>
            <pubDate>Tue, 19 Sep 2006 11:13:57 GMT</pubDate>
            <lastBuildDate>Tue, 19 Sep 2006 11:13:57 GMT</lastBuildDate>
            <image>
                  <title>lemurs - Google Noticias</title>
                  <url>
                        http://news.google.com/intl/es_es/images/news_res.gif
                  </url>
                  <link>http://news.google.es/</link>
            </image>
      </channel>
</rss>
```

11.2 Atom 0.3

```
<feed version="0.3" xml:lang="es">
      <generator>NFE/1.0</generator>
      <title>lemurs - Google Noticias</title>
      <link rel="alternate" type="text/html"
      href="http://news.google.es/news?ned=es&q=lemurs&ie=UTF-8"/>
      <tagline>lemurs - Google Noticias</tagline>
      <author>
            <name>Google Inc.</name>
            <email>news-feedback@google.com</email>
      </author>
      <copyright>&copy;2006 Google</copyright>
      <modified>2006-09-19T11:18:09+00:00</modified>
```

```
</feed>
```

11.3 Magpie RSS

A continuación se muestra un ejemplo de como se ha implementado el lector de *feeds* utilizando MagpieRSS.

```
<?php
require_once 'magpierss-0.72/rss_fetch.inc';
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<link rel="stylesheet" type="text/css" href="estils/estil.css">
</head>
<body>
<?php
if(!isset($_POST['q']))
{
?>
      <form action=<?=$_SERVER['PHP_SELF']?> method="post">
            Terme a cercar: <input type="text" name="q"/>
      </form>
<?php
}
else
{
      url = "http://news.google.es/news?ned=es&q=" . $_POST['q'] . "&output=rss";
      $rss = fetch_rss($url);              //Consulta de feeds google
?>
<br/>
<?php
      foreach ($rss->items as $item )        //Presentación resultados usuario
      {
            $desc = $item[description];
            echo "<div class='resultat'>$desc</div>\n";
      }
}
?>
</body>
</html>
```

12 Horas De Trabajo

- Inicio del proyecto: 15 diciembre 2005
- Fin del proyecto: 8 Enero del 2007

→ Número de meses dedicados: 11

→ Media de horas de trabajo semanal: 20

→ Total de horas: 880

→ Distribución de las horas:

- Análisis y Diseño: 165 Horas (19%)
- Estudio de tecnologías y herramientas: 271 Horas (31%)
- Implementación: 229 Horas (26%)
- Documentación:215 Horas (24%)

13 Agradecimientos

Especialmente a mi padre y mi madre por el apoyo moral.

Mis compañeros David Huerva y Rafa Recasens por la ayuda y consejos dados.

Por último mi amiga Mireia por escucharme y el tutor.